AFIT/GCS/ENG/98D-02

A Web-based Prototype for AFIT Edplan Administration

THESIS
Tien-Chen Lee,
Captain, Taiwan AF

AFIT/GCS/ENG/98D-02

Approved for public release, distribution unlimited

Approved for public release, distribution unlimited

AFIT/GCS/ENG/98D-02

An Web-based Prototype for AFIT Edplan Administration

Tien-Chen Lee
Captain, Taiwan AF

Approved:

Michael Talbert 11 Dec 98

Michael L Talbert, Ph.D., Major
Thesis Advisor
Date

Richard A Raines 11Dec98

Richard A.Raines, Ph.D., Major
Committee Member
Date

Henry B. Potoczny, Ph.D. Dec. 11, 1998

Henry B. Potoczny, Ph.D.
Committee Member
Date

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department or the U.S. Government.

A Web-based Prototype for AFIT Edplan Administration

THESIS

Presented to the faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Computer System

Tien-Chen Lee

Captain, Taiwan AF

December 1998

Approved for public release, distribution unlimited.

Acknowledgments

I want to thank my thesis advisor, Major Talbert, for his sound support, guidance, and much needed course corrections. I would like to thank my committee members, Major Raines and Dr. Potoczny for additional support and guidance on both my research and my studies at AFIT.

I would also like to thank Dan Dipiro, who provided the methodology I used in my thesis, for his assistance to my research. My appreciation and thanks also go to my sponsor couple, Michael L Kinane and his wife Estella Kinane, who have helped me in English writing skills and many other supports.

Table of Contents

Table of Figures

List of Table

AFIT/GCS/ENG/98D-02

Abstract

This document details the design, development, and evaluation of a prototype course registration and reporting system for the students and faculty of the United States Air Force Institute of Technology. The web-based system provides HTML-based client interfaces and Active Server Page server processes for interaction with the relational databases used to manage course and personnel data. The system prototype was developed following the "Engineering Software Components for Web-Database Access" methodology of Dipiro. A survey of modern web-based database access techniques is first provided and Dipiro's methodology is reviewed as background. The remainder of the document details the application of the methodology as a decision aid for decomposing system requirements into a series of user interaction and data access functions. Then, again following the methodology, an analysis of extant web-database access techniques is performed in the search for the most appropriate one. Next, the developed prototype system's functions are described and depicted via screen capture images. Finally the results of prototype evaluation via user feedback surveys are provided along with recommendations for future system improvement. Ultimately, this work stands as a validating test case for the Dipiro's methodology.

A Web-based Prototype for AFIT Edplan Administration

1 *Introduction*

In recent years, the technologies comprising the Internet have been developing rapidly, due mostly to the rising utility of the World Wide Web. The introduction of the WWW to the Internet has caused a great impact on not just the computer world, but also many individuals and enterprises. As the WWW becomes more useful and powerful and the prices of personal computers (PC) continue to drop, the number of users is growing rapidly. The WWW was originally used primarily for distributing static technical information, but today it is being used for more a much broader genre of tasks. Since it is so useful and powerful, it has triggered the desire of many different enterprises and organizations to make consumer-ready information (data) available through this powerful medium (WWW). Consequently, many technologies have been developed for web applications to make these enterprises' data in standing database management systems available via a browser-based interface to the WWW.

1.1 Problem Statement

While many Web-based software technologies have been developed and created, it has become more difficult for developers to choose an appropriate technology for their projects. Dan Dipiro [Dipiro98], who graduated from Air Force Institute of Technology (AFIT) March 98, offers a methodology to analyze the currently existing technologies for Web-based database access to help developers to pick the right one for their applications. His methodology focuses on essential factors pertaining to the development environment, client capability, and technologies available, which affect the design process and the capabilities of each Internet technology.

AFIT course Education Plan (edplan) Administration is routine but critical task for AFIT students and faculty. The edplan is used to record a student's projected and actual academic programs. Each new AFIT student is required to submit an initial education plan that lists all courses to be included in the academic program. The edplan is prepared in conjunction with the student's academic advisor. It helps faculty advisors ensure that the edplans, which students propose, satisfy the requirements for their programs. Usually, students will create the edplan either by using AFIT's edplan checker, an automated education plan checking program, or by typing manually. After students submit their initial edplan to their advisors, those faculty advisors enter the edplan into the school's database, which stores all data about AFIT faculty and students. Currently, students can not view and change their edplan by themselves. They have to go to see their advisor to view or change. Students also have to create the edplan checker input file to run the edplan checker program to see if they satisfy the program requirements. Current students have to create this input file by manually typing it themselves. Because the edplan checker program is written in Prolog which is very particular in file format, it is very easy for students to have typing errors and cause time wasting when running the file on the edplan checker program. Due to the problems described above, a web-based edplan prototype system will be developed to make the current process of handling these matters efficient and reduce the chance of manually typing errors. Additionally, because this application is in the same domain of Dipiro's methodology, I will implement the methodology to choose an appropriate Web-based technology and develop the Web-based prototype for AFIT Education Plan Administration.

1.2 Purposes of this Research.

The goals of this research are to implement Dipiro's methodology to analyze the existing web-based data access software technology and choose a good one to create a prototype edplan system. And also investigate the ways of the connectivity between database, such as Java Database Connectivity (JDBC), Open Database Connectivity (ODBC) and ActiveX Data Object (ADO). The attraction of the WWW motivates the design of a Web-based edplan function to help both the students and faculty to maintain the edplan easily. This application will provide an interactive page to let students input their initial edplan data as well as course credit category information, which doesn't exist in current ACES database. In particular, course credit category information is critical in ensuring specific graduate program requirements have been satisfied by the mix of courses taken. After the input data is generated, it will be sent to the prototype database directly. The data generated may be used to apply to other academic functions, such as an AFIT Form 69, an official record of the courses taken and the requirements they satisfy. The finished prototype system will be very useful for students as an "edplan on-line" interface, which helps them to generate and maintain an edplan more easily and efficiently. Consequently, this would save the faculty time currently spent doing this for students and it will also be useful to faculty for automated generation of AFIT Form 69 and it may be useful for other academic functions in the future.

1.3 Thesis Overview

Chapter 2 provides background information in the areas covered by this thesis, with a focus on some current database connectivity techniques (such as CGI, ODBC, JDBC and ADO), software technologies for web-base database access (Applets, ASP) and Dipiro's methodology. Chapter 3 describes the analysis and design of the project functions by implementing Dipiro's methodology. Chapter 4 discusses the implementation of Dipiro's methodology for the prototype system and the evaluation of the finished prototype. Finally, Chapter 5 contains findings, conclusions, and areas for future system improvement.

2 *Background*

In this chapter, background information is provided on the technologies and methodology discussed in this thesis. The first section provides the fundamental background of current database access techniques, such as Common Gateway Interface (CGI), Open Database Connectivity (ODBC), Java Database Connectivity (JDBC), Active Data Object (ADO), etc. The second section gives information about applets and Active Server Pages (ASP), two software technologies for web-database integration. The third section overviews a methodology for mapping web technologies to data access problems, proposed by Captain Dan Dipiro [Dipiro98], GCS 98M. The goal of this chapter is to give the reader idea of different technologies connecting to database through the WWW and to provide a foundational understanding of Dipiro's methodology.

2.1 Database Access Techniques

There are many methods of remote database access available today. Application developers can choose from the Common Gateway Interface (CGI) and Fast CGI, Open Database Connectivity (ODBC), Java Database Connectivity (JDBC), Object Linking and Embedding Database (OLE DB), Active Data Object (ADO), Data Access Object (DAO), gateways, proprietary Application Programming Interface (API), and no doubt others in various stages of implementation. The industry has evolved from monolithic, mainframe applications to client-server and distributed computing [North97]. The computing world has been developing in every aspect. The more it develops the more data needs to be handled. Due to the frequency of data exchange between corporations or individuals, database access has become a critical mission issue. As a survey of extant technologies, some background knowledge of database access techniques is provided.

2.1.1 Common Gateway Interface (CGI)

The CGI is a standard for interfacing external applications with information servers, such as web servers. CGI programs are executed in real-time, which allows them to dynamically process information, and also run their applications in processes isolated from the web server. Thus CGI overcomes some of the limitations of otherwise static web pages by offering the ability to dynamically create HyperText Markup Language (HTML) pages in response to user requests, fielded by the web server. Many early solutions to web-based data access relied on the CGI and server-side applications [Dipiro98]. The disadvantage of CGI applications is that a web server starts up an instance of the CGI program for each user request. Some web servers are equipped with an additional set of Application Programming Interfaces (APIs), such as Microsoft's Internet Information Server API (ISAPI), Netscape's Netscape Server API (NSAPI), or Apache's ASAPI, which don't use the "process-per-request" model.

While these APIs provide similar functions, they are vendor-specific and often incompatible. Fast CGI, a faster and open interface from Proprietary APIs, is a new, "open extension to CGI that provides high performance for all Internet applications without any of the limitations of existing Web server APIs" [FastCGI]. Fast CGI is simple because it is actually the extended CGI and performs much better. Since FastCGI is an open specification, you can run it without depending on software, which solves the problem of vendor-specific APIs.

2.1.2 *Open Database Connectivity (ODBC)*

ODBC was developed by Microsoft in 1992. ODBC is an API for programs that use Structured Query Language (SQL) to access data. The main purpose of the ODBC interface standard is to enable applications to access data from a variety of database management systems (DBMS) using a standard SQL-like interface. ODBC allows for maximum interoperability and alleviates the need for independent software vendors and developers to learn multiple APIs. ODBC provides a universal data access interface and application developers can allow an application to concurrently access, view, and modify data from multiple, heterogeneous databases.

The ODBC interface supports the following:

- A library of ODBC function calls: These function calls allow an application to connect to a DBMS execute SOL statements, and retrieve results.
- SQL syntax support: The SQL syntax is based on a Call Level Interface from the SQL Access Group (SAG)(now part of X/open) and X/open that advanced to become the international standard CLI for SQL-92.
- Error standard: It provides a standard set of error codes.
- Connection standard: It provides a standard way to connect and log on to a DBMS.
- Data type standard: It offers a standard way to represent data types [Hettih97].

When an application tries to communicate with a DBMS, it will use ODBC interface functions to perform many tasks, such as requesting a connection or session with a data source, sending SQL requests to the data source, defining storage areas and data formats for the results of SQL requests, processing errors and requesting commit or rollback operations for transaction control, then finally terminating the connection to the data source. An application can also connect to multiple data sources concurrently. After the data source is connected, the application can then perform the tasks described above.

The architecture of ODBC can be classified as four components as follows:

- Application: Performs processing and calls ODBC functions to submit SQL statements and retrieve results, also known as an "ODBC Client".
- Driver manager: The driver manager is provided by Microsoft and it is a Dynamic Link Library (DLL), which allows to load drivers dynamically when they are needed. It loads and unloads drivers, performs status checking, and manages multiple connections between applications and data sources.
- Driver: A driver process ODBC function calls, submits SQL requests to a specific data source and return results to the application. Any ODBC client (application) can access any DBMS for which there is an ODBC driver.
- Data source: Data source is the back-end, which consists of the data the user wants to access and its associated operating system, DBMS.

The figure 1 shows the architecture of ODBC.

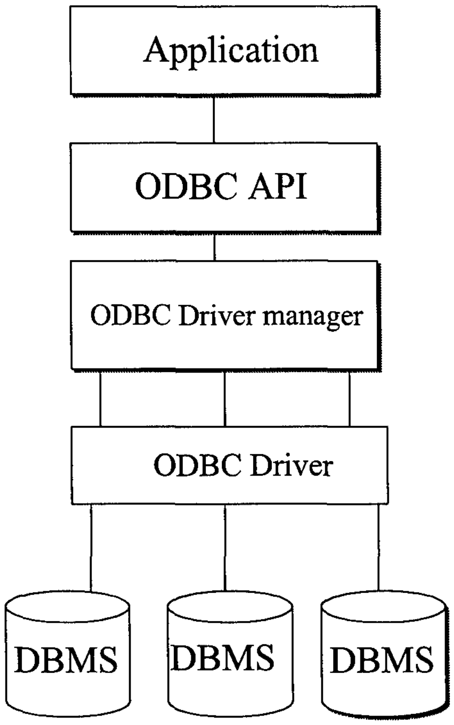


Figure 1. The Architecture of ODBC.

Most of the main tasks are done by DBMS-specific drivers. In response to ODBC function calls from an application, a driver establishes a connection with and submits requests to a named data source. It also translates data to or from other formats, returns results to the application, and formats errors into standard error codes and returns them to the application. The ODBC standard has been widely accepted as a vendor-neutral API to access heterogeneous SQL databases. The capabilities of ODBC make the data connection easier and help application developers to concentrate more on the core functions of the applications.

2.1.3 *Java Database Connectivity (JDBC)*

Before description of JDBC, some background information may be useful to know. The object-oriented paradigm of programming has swept through the computer science industry over the past five years. Java is one of the most popular and powerful object-oriented languages modeled after C++. Java was released by Sun Microsystems in the summer of 1995. One of the foremost features of Java is its hardware independence. Java source code is compiled into bytecodes, which are platform independent [JAVA96]. Due to Java's platform independent feature, it is an ideal language for Internet applications since it can easily interface with the various kinds of computers and operating systems in use throughout the Internet, such as UNIX, Microsoft Windows and Macintosh. Java programs that are embedded in World Wide Web pages are referred to as "applets", while stand-alone programs are called "Java applications". Before Java, users viewed HTML-based pages using Web browsers, such as Internet Explorer, or Netscape. These pages contain text, pictures, audio and video clips, and other multimedia files. With new "Java-enabled" browsers (such as Netscape 2.0 and higher), users may visit a web page with an embedded Java applet, which is downloaded from the web server to the client (user's personal computer) and executed on that user's machine. This allows web pages to perform many complex tasks such as animation, interactive games and database access. In addition, Java supports multithreading and multiprocessing, making it suitable for database applications that involve large numbers of concurrent users.

Due to the popularity of the Java language, Sun released Java Database

Connectivity (JDBC) in June 1996. JDBC is an SQL API that includes Java classes, a driver manager, and loadable drivers. JDBC shares a common objective with Microsoft ODBC, which provides connectivity to SQL-based databases and provides developers with an open standard for hooking their databases to the Web. JDBC and ODBC are both based on the Call Level Interface (CLI) from X/open [SUN96]. Developers familiar with ODBC should feel comfortable with JDBC's environment, since JDBC shares some common classes, such as database connections and callable SOL statement classes. The JDBC's architecture is similar to ODBC's, except that JDBC can also use a JDBC-ODBC Bridge to connect to ODBC drivers. The JDBC-ODBC Bridge, jointly developed by JavaSoft and Intersolv, maps JDBC logic into the appropriate ODBC function calls. The remote DBMS (with the appropriate ODBC drivers) receives these calls as they do from any other ODBC client, and executes them. Applications written in Java, including Java applications and applets, can use JDBC to access remote databases. Figure 2 shows the architecture of JDBC and the relationship between applications, JDBC and DBMS.

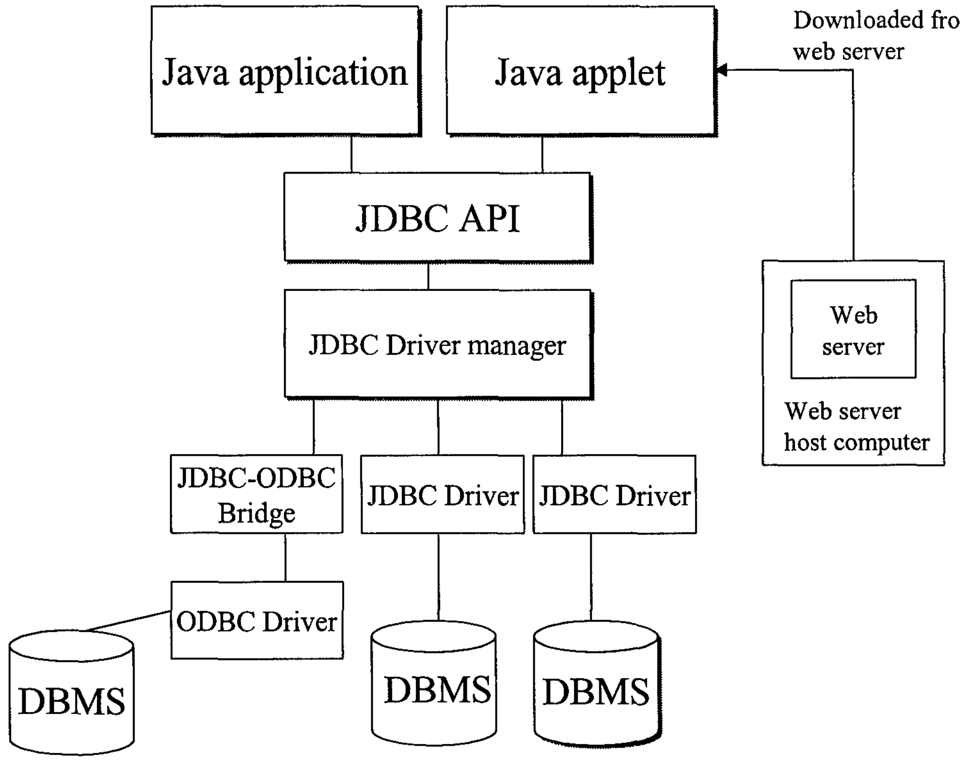


Figure 2. The Architecture of JDBC and the relationship of JDBC and DBMS.

With the growing popularity of Java, the Java language is evolving very quickly, and new Java products are appearing each day. In the future, JDBC may be the connectivity API of choice for many database systems.

2.1.4 ActiveX Data Objects (ADO)

ActiveX Data Objects (ADO) is Microsoft's strategic, high-level interface to all kinds of data. ADO is built on top of the Object Linking and Embedding applied to databases (OLE DB)[1] model from Microsoft. The main purpose of the ADO is data access and manipulation. ADO is the successor to both Data Access Objects (DAO)[2] and Remote Data Objects (RDO)[3]. ADO combines the best of previous data access methods with an object-based standard, and includes the capability of DAO and RDO and extends their reach to provide data access for the Internet using the OLE-DB model.

ADO includes several objects, three of which are fundamental for interacting with a database. Those three fundamental objects are:

- The Connection object: The connection object is used to create a connection to a data source. Once the connection is established, you can issue a query against the database and get results back.
- The Command object: The command object enables you to specify a specific command that you are going to execute on a database. This object is useful when issuing a data manipulation query, such as an insert.

[1] OLE DB was designed to give the developer access to a wide range of data storage systems. ODBC only provides the interface to relational data stores based on SQL. The idea of OLE DB is not to replace ODBC, but to extend it.

[2] DAO was developed to encapsulate database functions and operations within the context of an object. DAO provides access to ODBC-compliant databases.

[3] RDO was the successor to DAO and provided a better solution for ODBC database access and extended the reach of these objects to the server.

- The Recordset object: The recordset object is used to manipulate the records, or rows within your database tables. This object uses the cursor for a query to help you to traverse the result table [Hettih97].

ADO is designed to be an easy-to-use application level interface for today's developers. Microsoft implements ADO specifically to provide data access across the Web and they also publicly stated that ADO eventually will replace the company's current data access models, including DAO and RDO. Thus ADO won't be just for Internet / Intranet data access, it is the Microsoft data access model for the future [Litwin98].

2.2 *Software Technologies for Web-based Applications*

The software technologies for web-based data access can fit into two categories with respect to the client-server model. The two categories contain technologies used to create code designed to run on the client (client side) and code designed to run on the server (server-side), respectively. This section provides brief introductions of two different software technologies for both client side and server side.

2.2.1 *Applet (client side)*

Applets are executable programs stored on the server and created with the Java programming language from Sun Microsystems. Applets are embedded in a web page and execute within the context of a browser. The browser must support Java to be able to execute the applets. When a user requests a certain web page, the HyperText Markup Language (HTML) is sent with the applets from the server to the browser on the client machine. The browser detects the applet by discovering the <APPLET> tag within the HTML document, then it interprets the compiled Java bytecodes with the Java runtime interpreter (most browsers support Java runtime interpreter nowadays) and executes the applet program.

These applets can provide various functionality from multimedia to spreadsheet applications. Currently applets have been used to perform complex tasks such as animation, database access etc.

2.2.2 *ActiveX Server Pages (ASP) (server side)*

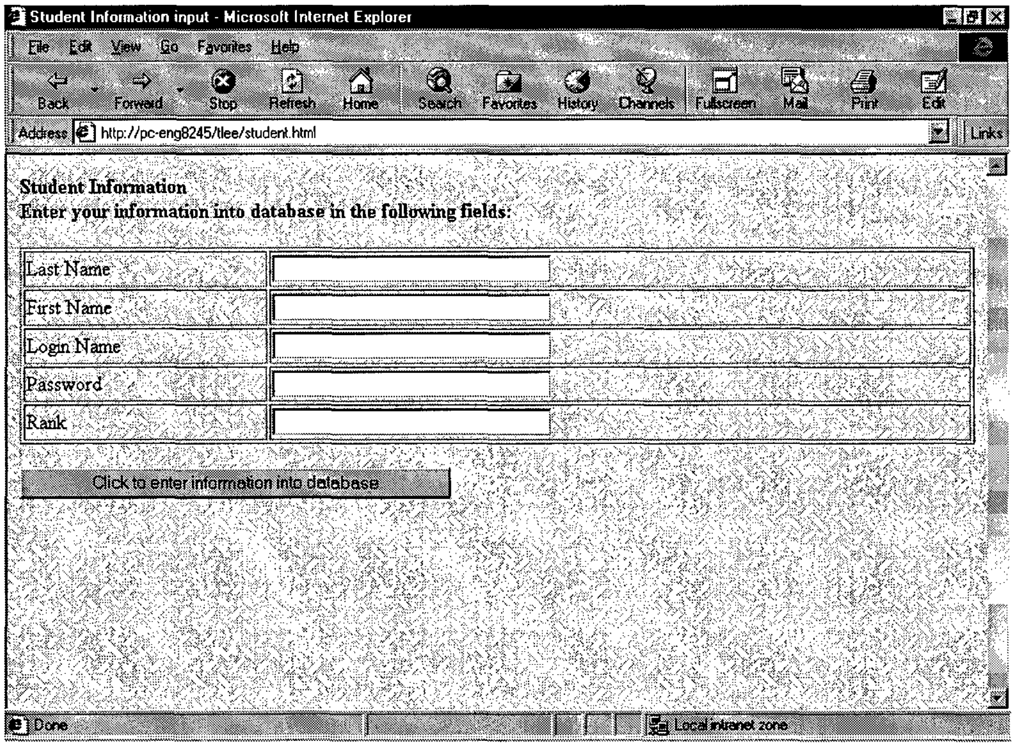
ASP is Microsoft's most recent Web server technology, and is designed to make it easier for developers to create sophisticated Web applications. Microsoft originally created ASP as the integral server processing component of their web server, Internet Information Server (IIS). Now the Microsoft Personal Web Server includes ASP as a component, too. The statement above does not mean that ASP can only be used by Microsoft web servers though. In fact, ASP can work with any other web servers, including Netscape's web server.

ASP is a server-side scripting environment that includes built-in objects and components, such as ADO, which enable a degree of interactivity and greatly simplify many complex tasks. ASP applications can combine HTML, scripting language (scripts), and ActiveX server components to create useful Web-based applications. To make it clear, a simple ASP application is presented next. The code in Figure 3 is simply an HTML displaying a student's information input form.

```
<html>
<head>
<title>Student Information input</title>
</head>
<body>
<p>Student Information<br>
Enter your information into database in the following fields:</p>
<form action="students.asp" method="POST">
 <table border="1" width="100%">
  <tr><td>Last Name</td>
    <td><input type="text" size="30" name="lname"></td></tr>
   <tr> <td>First Name</td>
    <td><input type="text" size="30" name="fname"></td></tr>
   <tr><td>Login Name</td>
     <td><input type="text" size="30" name="login"></td> </tr>
    <tr><td>Password</td>
     </tr> <td><input type="text" size="30" name="password"></td></tr>
    <tr> <td>Rank</td>
      <td><input type="text" size="30" name="rank"></td>
 </table>
 <p><input type="submit" name="B1" value="Click to enter information into
database"></p>
</form>
</body>
</html>
```

Figure 3. HTML code for student.

The output of that HTML code is shown in Figure 4 below.


Student Information input - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit
Address http://pc-eng8245/tlee/student.html
Links
Done
Local intranet zone

Student Information
Enter your information into database in the following fields:

Last Name

First Name

Login Name

Password

Rank

Click to enter information into database

Figure 4. Student information input HTML page.

The following code in the Figure 5 does the main job. This code is a combination of Visual Basic script, HTML and ADO (Connection Object). The Connection object of ADO creates the connection to the target database and "if …then" syntax of VB scripts is used to build the logic of this application. Scripts delimiters are used to separate the scripts and HTML. This application simply shows how to insert data into database by using ASP.

```
<!--#include file="IASUtil.asp"-->
<html>
<head>
<title>Students' Info</title>
</head>
<body>
<%
Set SDB = Server.CreateObject("ADODB.Connection")
SDB.Open "Students"
query = "SELECT * FROM Students WHERE
LastName = '" & Request("lname") & "' AND FirstName = '" & Request("fname") & "'"
Set Findquery = SDB.Execute(query)
If CheckRS(Findquery) then
%>
<h2> A student with that last name and first name  already exists!</h2>
Click back and try again.
<%
Else
query1 = "INSERT INTO Students (LastName, FirstName, LoginName, Password, Rank)
VALUES('" & Request("lname") & "','" & Request("fname") & "','" & Request("login") & "','
" & Request("password") & "','" & Request("rank") & "')"
Set InsertQuery = SDB.Execute(query1)
%>
<h2> Student's info entered into database.</h2>
<%
End If
Findquery.close
SDB.close
%>
</body>
</html>
```

Figure 5. ASP code for inserting student input to DB.

Unlike CGI, which runs isolated from the web server, ASP runs processes on the server, and is multithreaded and optimized to handle large numbers of users [IIS4.0]. Scripting is really how ASP holds all the pieces together and controls the program flow of the web page. ASP 's database access component uses ADO to access and manipulate information in a database. Figure 6 shows how ASP interacts with databases.

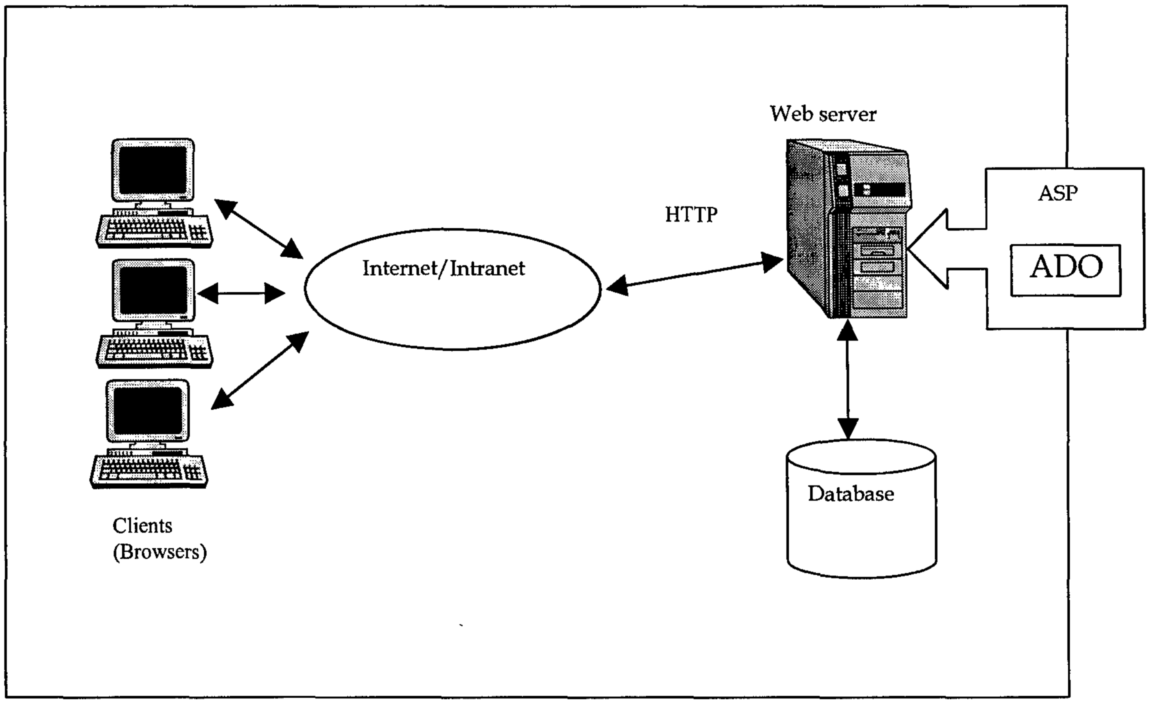


Figure 6. ASP application interacting with Web-database.

ASP is specially designed to make it easier to develop interactive Web applications and to work together with Windows technologies such as ActiveX, ADO, and ODBC. Therefore ASP would be a very good software technology to choose if the developers intend to use Microsoft products.

2.3 *Methodology of Dan Dipiro*

Dan Dipiro graduated from AFIT/ENG school with a Masters of Science in Computer Systems in 1998. The goal of his thesis research was to provide a methodology that could be used in the analysis of a web-based data access enterprise to aid the developer in choosing the right software technologies to implement their data access services. His methodology consists of three main steps - Development Environment Analysis, Component Analysis and Design, and Function Implementation. The first step of this methodology, Development Environment Analysis, includes the analysis of three environment factors. This information and knowledge of the capabilities of each Internet technology are actually enough for a developer to make a choice. Thus this section will focus more on the first step and describe the other steps in more general terms.

2.3.1 *Development Environment Analysis*

This step describes how certain aspects of the overall system environment can impact or influence the technologies chosen for implementing data access. The first of those aspects to consider is the clients who will access the data sources, the second is the existing database architecture, and the third is the availability of development and maintenance resources. In this methodology, the overall project is ultimately defined as an interoperating aggregation of individual components, each performing a unique function.

2.3.1.1 Client Environment

The client environment can be defined in two classes of clients for a typical web-based scheme. The first consists of internal clients who are members of the organization owning the data sources. The second, external clients, are those clients who exist outside the organization, but who need to access the data sources.

Internal clients

Usually, information about the hardware and software platforms that the internal clients use, and the number of these clients, are available to the developers. Thus the developers can have more accurate task-to-technology mapping in choosing the software technologies for their projects. The developers can always select the best technologies to fit their internal clients' platforms to increase the application capability and provide more optimized run-time performance. Another positive aspect of internal clients is that they are most likely to have a high-speed connection to the server, since they usually are on the same network or local area network (LAN).

External clients

The application can also be designed for a specific number of approved external clients or an unlimited number of potential external clients. The number of external clients can have a great impact on a web server's performance. This is because as the number of concurrently connected clients increases the server's processing load, and can cause it to become overburdened if it does not have sufficient processing power. Since a server typically has only one network connection, response time can also be affected by a communications bandwidth bottleneck. Thus the developers must project and plan for the potential impact of the increased external client load on both network bandwidth and server performance.

The software and hardware used by external clients is generally hard to predict. Therefore, the designer must always consider portability to the expected cause of client platform capabilities when designing components to be accessed externally. For example, the impact of the choice to use a proprietary technology could cause unpredictable results. Some clients may experience no degradation in component functionality, others may get limited functionality, and some can have total loss of functionality [Muell97]. For example, consider an applet created in the Java programming language. A client using Sun's HotJava browser will see the optimum result that the designer expected. A Microsoft Internet Explorer client may see the application well, but some functions may not work correctly. Some other clients with other browsers may even experience the worse case, such as not seeing the component at all. Developers must also consider the external client's network connectivity when designing components to the system, since a long download time really degrades the perceived overall performance of the system. In short, complete information about external clients is unknown and cannot be known with any certainty. Therefore, the component that they interface with should be designed for portability and of a reasonable size to optimize download performance.

2.3.1.2 Analysis of the Existing Data Source Architecture

The second step to be considered when choosing technologies for web-based database access component design is the existing database architecture. This includes any existing database management systems, their ability to handle an increased client load, and the physical location of the data sources.

Existing Database Management Systems

Two key areas of the existing DBMS are focused on in this step. The first is the multi-user capability of the existing database management software. The ability of a DBMS to service remote web clients will depend greatly on whether the DBMS implements a single or multi-user environment. The web clients accessing a database through most interfaces are treated the same as local clients by most DBMS. Thus using a single-user DBMS, or one that can only handle a small number of concurrent users, will greatly reduce the overall performance for the web-based application. The second area that will influence any design is the existence of any original equipment manufacturer (OEM) or third party development tools. These tools allow the rapid graphical-based design of web forms and reports, and also automatically generate the code necessary to compile these applications. The major benefit of these development tools is their tight coupling with the target data source. The tight coupling between the code created by the tool and the DBMS ensures that the web-components will be more functional and have fewer bugs. A negative aspect of these tools is that they tend to be very expensive if not bundled with the DBMS. Some databases can only be accessed via the web through a proprietary interface. This will limit a design to the methods supported by that DBMS. Many DBMSs such as Microsoft Access can communicate with clients through many ways such as ODBC or through a JDBC-ODBC bridge. Using these types of DBMSs will make it more flexible for designers to choose the technologies for applications.

Location of Data Source

The second area of the existing DBMS architecture that deserves consideration is the physical location of the data sources. There are two cases, the first of which is that the data resides in a single database or in multiple databases on a single server. The second is when the data to be accessed resides on different machines. The second case is the primary concern. A web application needing to access data on several different machines will affect a design by increasing its complexity and introducing potential security and performance factors. Attempting to access several machines can also violate the browser's security constraints. Usually most web browsers only let applets connect to the machine from which they were served the web page. According to Dipiro's methodology, to avoid the difficulties of this case, a three-tiered approach should be used. In this approach, a server process on one machine handles the interaction with the databases by receiving requests from the client, and then executes them on the appropriate database servers. Using the three-tiered approach can avoid the security restrictions imposed by the client's browser and also maximize the performance of data access by residing on a server with a fast communication path to the database.

2.3.1.3 Existing Resource Environment

The third factor to consider in this step consists of the available resources of time, money, and personnel. This section provides information to aid the designer who is limited in at least one area of those three aspects.

When time is a limiting resource, a designer should seek to create components that are simple, clearly documented, and that maximize code reuse whenever possible. Also the designer should be familiar with the capabilities and drawbacks of each technology in order to help decide which best suit design goals and environment constraints.

For funding resource limitations, the price information of web server and development tools are available to help designers make the right choices within their budget. The impact of funding resource limitations is that it can restrict the tool choices of the developers.

In terms of development personnel limitations, if there are plenty of personnel for analysis and development, a more in-depth analysis can be done of the competing technologies which can also reduce the overall time needed for component development. If the developer is limited in personnel, he should at least seek to learn the capabilities, drawbacks, and communications capabilities of the technologies to be considered before analyzing and designing the components. Dipiro recommends that the developer may choose a technology that he is familiar with or easy to learn, or one that develops a web component through a development tool included with their DBMS.

2.3.2 *Component Analysis and Design*

The second step of Dipiro's methodology takes a structured approach to creating the individual components that will comprise an overall web-based access project. This step guides the developer in deciding which technologies will meet the specifications of each individual component. A component here is considered an application or combination of applications, that implements one clearly defined function. An example component, "to reserve a library study room", is used for analysis and design. The example will involve five sub-components. The first sub-component obtains the desired time slot from the user. The second queries a database to find out which rooms are available in that period. The third and fourth display the results to the user and handle a room reservation request respectively. The fifth sub-component processes, then displays, the result of the request back to the user.

Functional Analysis Overview

An object-oriented (OO) functional modeling diagram technique called a Data Flow Diagram (DFD), which is useful to depict the component's overall function [Rumba91], is used to perform a functional analysis of the component to be designed. In the DFD, the process is represented as a circle containing a description of the process. Arrows leading into a circle denote an input to the process. Arrows leading out denote output. Any persistent data stores are represented by the database name with a line above and below. Figure 7 shows the top-level DFD for the example problem.

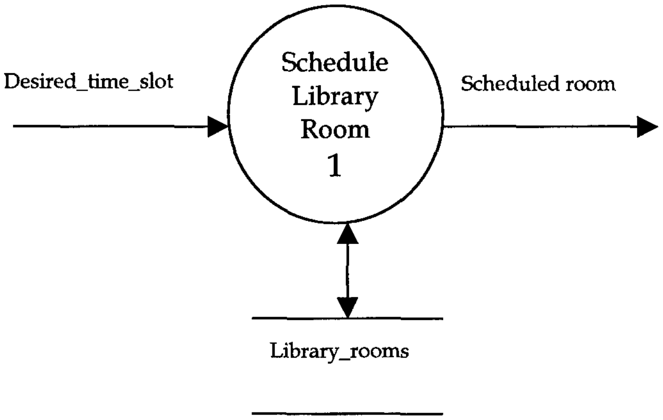


Figure 7. DFD for Schedule Library Room Component.

The component then can be decomposed into any necessary sub-components. The level to which decomposition is performed is at the designer's discretion.

Component Functional Decomposition

Functional decomposition is used in this methodology to determine the sub-functions that need to be built, and those sub-functions may be the applications themselves. When decomposing the processes of a component into sub-processes, it is broken into smaller sub-components, whose function is a sub-task of the overall component process. For the room reservation example, five sub-components are required for implementation. Figure 8 shows the lowest level DFD of the room scheduling component. The first function gets the desired start and stop time from the client. It then sends those times to the second function that must determine which rooms are available during that time slot. That process, in turn, sends the client the results of the query for viewing and reserving a room, if desired. If the client requests to reserve a room, the fourth process handles the request and sends the results to the fifth to be displayed for the client.

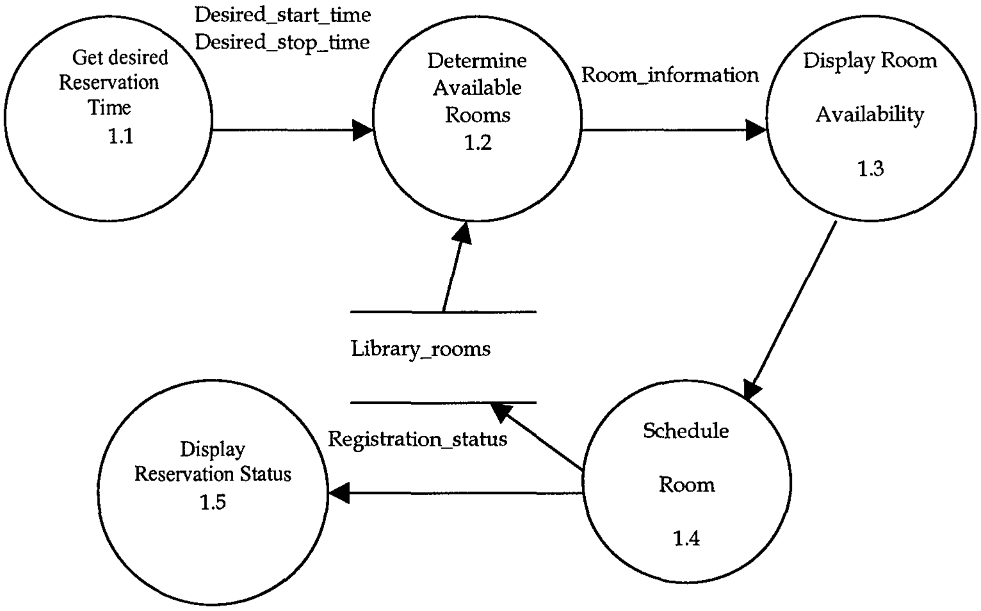

Get desired Reservation Time 1.1
Desired_start_time
Desired_stop_time
Determine Available Rooms 1.2
Room_information
Display Room Availability 1.3
Library_rooms
Display Reservation Status 1.5
Registration_status
Schedule Room 1.4

Figure 8. Lowest Level DFD of Room Scheduling Component.

2.3.3 *Function Implementation*

This step is to take the fully decomposed component and analyze the possible technologies that can be used to implement each sub-component process. The developer may choose to implement any or all sub-components in any of the existing web technologies rather than just a single one. It is important for the developer to be familiar with the methods in which the different technologies can communicate with each other. This is because the choice to implement a sub-component in a certain technology may limit what technology the developer can use for adjacent sub-components. For example, a static HTML page can only originate from a web server process (such as ASP, CGI) and can only send output to a sub-component that can receive an HTTP request. This step provides the information about which technologies can communicate and which can't. For example, an applet is capable of making calls to a database over the internet via JDBC or ODBC, but a stand-alone application can't interface with other components through the HTTP protocol, and only web server processing (CGI, ASP) can receive input from, and output to, HTML documents using the HTTP protocol. The main part of this step, to use the library example from the last step, is to make assumptions concerning the design specification and demonstrate how to implement the project based on that specification. The assumptions made for this example are shown in the following:

- The component will be accessed by external clients only
- The client uses a variety of hardware platforms running a similar variety of operating systems (Unix, Windows, Macintosh, VMS, etc.)
- Some users may have low bandwidth connections (dial-ups)
- The client interface should look the same on all platforms
- The user should be able to print the reservation confirmation for their records
- No database architecture changes are necessary to handle additional clients

A starting point should be chosen first to start implementing the project. Any sub-component which is constrained by one or more of the three environment factors can be a starting point. Usually, a starting point occurs in either a user interface, or a database access sub-component. From the assumptions and specifications of the example, due to the heterogeneous environment of the clients and the potential for low bandwidth connections, the starting point should be at the user interface. In the example static, HTML pages are chosen for all interfaces with the user (client). This maximizes the sub-component portability (visible on all browsers) and minimizes its bandwidth requirements (no downloaded code sent to client). Therefore, Figure 8 Sub-Component 1.1 (Get Desired Reservation Time), 1.3 (Display Room Availability), and 1.5 (Display Reservation Status) will all be HTML pages. Since HTML pages can only generate HTTP requests, and only a web server process can generate an HTML page, a web server process technology (such as ASP) was chosen to implement Sub-Component 1.2 (Determine Available Rooms) and 1.4 (Schedule Room).

2.4 *Summary*

This chapter presents an overview of four different database connectivity technologies (CGI, ODBC, JDBC, and ADO), two web-based database software technologies (Applets, ASP) and Dipiro's methodology. The techniques and technologies offered in this chapter are important to know to create the web-based data access applications. The knowledge of those technologies is used in conjunction with Dipiro's methodology as the foundation in the Chapter 3 to determine the technologies for building the target project.

3 *Design of the prototype system*

This chapter describes the application of Internet technologies to build a prototype system for the problem described in chapter 1. Following the methodology of Dipiro [Dipiro98] it examines the capabilities of each Internet technology, with the goal of choosing the appropriate technologies to accomplish the specific web-based data access project. The reason using Dipiro's methodology is because the problem we are going to solve is in the same domain with the methodology. Therefore his methodology should be good to perform for this project. This chapter also describes the processes and the importance of building the prototype system and provides a way to validate the finished product.

3.1 Dipiro's Methodology for choosing Internet technologies

As mentioned in chater2, Dipiro's methodology consists of three main steps, but in fact only the first two steps are implemented here to help determine the appropriate technologies and design the project in this chapter. The third step will be used to help implement the design of the prototype system. This section examines the applicability of the three development environment factors described by Dipiro. The edplan application described in chapter one is analyzed by looking at the possible clients who will access the data source, the possible database architecture, and the available development resources for building the application components.

3.1.1 Analysis of the client environment

According to Dipiro's methodology, when analyzing a development environment, the client environment is the first and most critical factor to consider, because the whole purpose of the proposed application is to service clients. In Dipiro's methodology, the potential clients of any web-based data access fall into two categories, internal and external clients.

3.1.1.1 Internal clients

For the edplan application, internal clients will be the AFIT/ENG faculty and AFIT/ENG students who use the Internet at school. Based on the information that EN secretary offers, the approximate number of internal clients will be 30~70 (currently AFIT/ENG has 30 faculty and the average number of new ENG students is about 40). The small number of clients in this category ensures that any server-side processing applications will not overload the web server. All internal clients' operating system, hardware and web browser platform are known or predictable. In AFIT, all computers are IBM compatible PC-based computers. Each computer runs the Microsoft® Windows®95 operating system and has Microsoft® Internet Explorer (at least version 3) as its web browser software. For the platform homogeneity, we might choose Microsoft technologies, such as ADO, ASP (see chapter2 section2.1.4 and 2.2.2), to build the target project. Additionally, all internal clients are connected to the web server over a shared 10MBps LAN. This fast connection ensures that applications created for internal clients will not suffer any significant performance loss due to network traffic.

3.1.1.2 External clients

In general, precise information on external clients can not be obtained with certainty. Thus, the developer must make estimates of external client platform possibilities in order to design components for external client use. In this case, since we will just use the AFIT/ENG as a test bed to build a prototype for the real system, the number of the external clients can be determined quite accurately. The external clients who will be interested in accessing the database are the AFIT/ENG students, which are approximately 80 (including lower and upper classmen, and doctoral students). The server is adequate to serve this expected loads imparted by this client pool. But if we try to build the application for a real system, more effort should be taken to approximate of the number of the external clients and make estimates of their possible platforms. Those can aid the developer in determining the necessary level of component portability and to ensure that any components designed for external use will not over load the server. Most of the external clients will typically connect through a dial-up

Internet connection (kilobits per second), which is orders of magnitude slower than the LAN. Furthermore, various different operating systems and hardware platform can be expected, but as long as they are able to establish a network connection and run web browser software (most computers support these), no system compatibility problems are to be expected.

The external client's browser software is also unknown. Even though there are not many browser choices available, current browsers vary greatly in some key aspects. For example, older browser software may not be able to run client-side code such as JavaScript, applets or ActiveX controls. Therefore components for external clients should be designed with technologies that support the largest possible subset of available browsers [Dipiro98]. Since all browsers support standard HTML, HTML documents are very portable for presenting information to the client. Thus, HTML is quite a good choice to building the sub-component for the project. It is popular today to offer the client the choice of scripted, framed, or plain HTML information delivery. For the purposes of this research, HTML-only presentation is employed.

3.1.2 Analysis of the existing database architecture

All the data about AFIT students is currently stored in the ACES database. The edplan prototype system needs only a few tables from the ACES, such as the Course, Course_description and Course_offering tables. Since in this research we are to build a prototype for a future enhancement to STARS system, we will transform the Oracle tables into smaller scale tables in Microsoft Access for demonstration purpose. Thus, Microsoft Access will be the DBMS for this application. The transformed data will be located and maintained on a single PC, which will be used as the web server for the edplan application. Because the data and server reside on the same machine, the application can access data with greater speed by eliminating network communications traffic. However, the DBMS and the web server have to share the memory and processor, which can result in lower overall performance as the server's workload increases.

3.1.3 *Analysis of development resources*

This section describes the effects of the availability of development resources, including personnel, funds and time. For this edplan application, it can be a one-man project or a group project. The product of the one-man may not be very robust, since some points of the design may be missed which may cause the application to not function well. A group project would be more robust, since every aspects of the design phase can be more done thoroughly. Unfortunately, time and personnel resource limitations will restrict the edplan application to a one-man project.

Naturally, a one-man project will increase the time required to develop the application. When the developer has a time constraint, existing code should be reused as much as possible to simplify application development and to ease code maintenance. Additionally, funds for this prototype are not currently programmed into the AFIT budget. Thus, due to the lack of personnel, time, and funds, we would certainly seek technologies which are inexpensive and easy to learn, to help ensure rapid but reliable prototype development. As described in chapter2, ActiveX Server Pages is easy to learn and requires no compilation. An ASP application can be written with any text editor, and compiled by the web server being accessed. The only requirement for developing an ASP application is to have a Microsoft Internet Information Server (IIS) or Personal Web Server (PWS). The Personal Web Server was designed to allow development of applications for use on a single machine. The limitation of PWS is that the application won't be able to support a high volume application with a large number of users, but it is suitable for the proof-of-concept intention of this research.

Based on the analysis above, the Microsoft Personal Web Server is chosen as the web server and ASP will be used as the technology for the project. ODBC is also one of the requirements for ASP applications, and is a Microsoft compatible product as well. So we anticipate ASP and ODBC inherently working together with great native speed. Therefore the ODBC technology is selected for the real connecting job to access and manipulate the target database.

3.2 *Prototype system analysis and design*

The overall prototype system, which will be developed, can be divided into six primary components. This section presents an analysis and design of each component. The requirements for each component will be briefly described and then the component will be analyzed using the functional decomposition technique from Dipiro's methodology.

3.2.1 *Component 1 (Student information input function)*

As mentioned earlier, the real world data is stored in an Oracle DBMS. Several tables are required for this case including Course, Course_description and Course_offering. After negotiating with the Database adminstror (DBA) in charge of that relational database, those tables will be copied from Oracle and transformed into Microsoft Access. Due to security issues, student personal information including e.g., Social Security Number (SSN), are not allowed to be copied. Thus some student input or data fabrication is needed for this prototype.

This component provides a form page to help students to input some of their personal information to the database. The information that students input will make them qualified to input their edplan data. The finished component should be able to let students input their data in an easy fashion, using aids such as drop-down menus and other ActiveX data entry controls. Students should be able to view and update their personal data as desired. A student should then have the ability to input their edplan data.

3.2.1.1 Functional Decomposition of the Student information input Component

The functional decomposition technique from Dipiro's methodology is used to depict the component's overall function of the system being developed. Component 1 requires several pieces of personal data from each student. The input data are the student's last name, first name, Social Security Number (SSN), program type (ex: GCE or GCS), program year (99M or 2000M), login name, academic advisor and military rank. All the data sources for the prototype are from the Edplan database created in Microsoft Access, which includes several tables, such as Course, Course_description, Eddata, Faculty and Students tables. The input information will be sent to the Students table in the Edplan database after which the student will be registered as a user, and allowed to input edplan data. Figure 9 shows the data flow diagram (DFD) for the student input component.

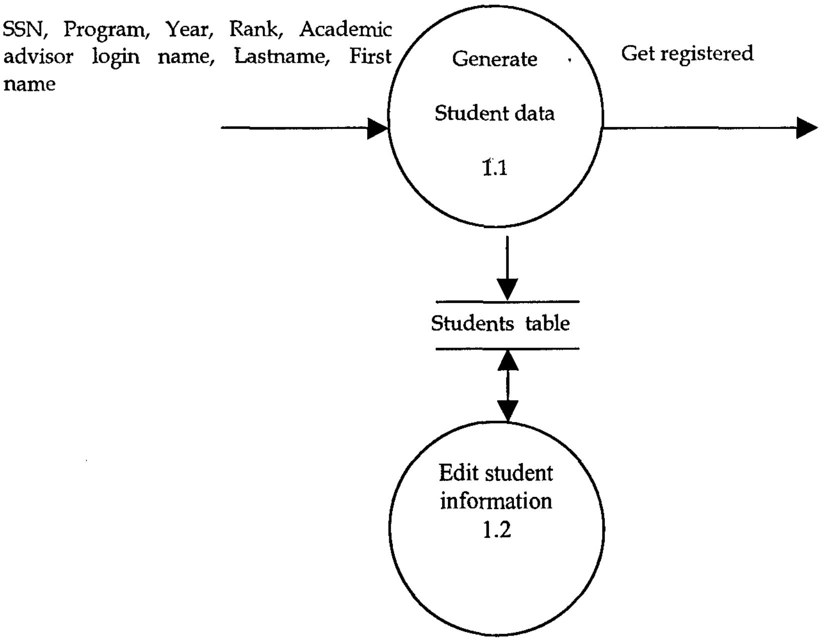


Figure 9. DFD for Student input component.

In order to input and view data into database and from database, the component must query the database to insert and update the data. The student's last name and last four digits of their social security number will then be used as inputs for authentication to this component and the subsequent components. A "cookie" (non- persistent or persistent) is created in conjunction with the registration event to track the identity of the student for the subsequent functions. This sub-component allows them to view or change their personal data. Therefore the component can be further decomposed into the function shown in Figure 10.

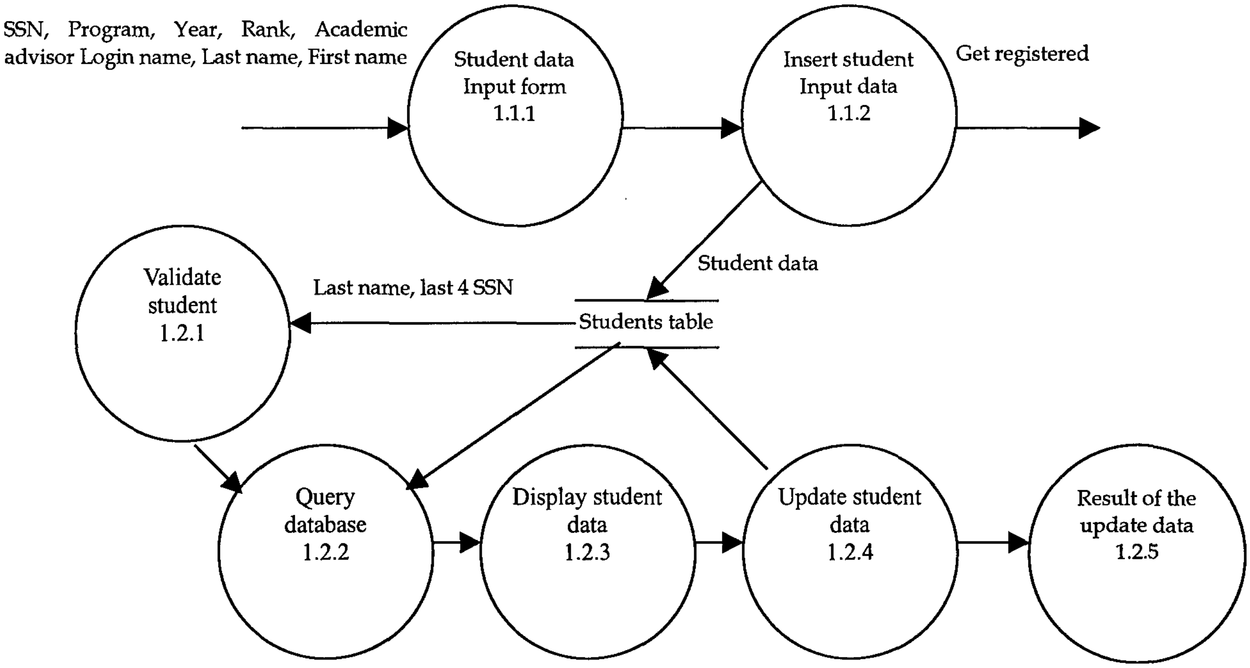

SSN, Program, Year, Rank, Academic advisor Login name, Last name, First name
Student data Input form 1.1.1
Insert student Input data 1.1.2
Get registered
Student data
Students table
Last name, last 4 SSN
Validate student 1.2.1
Query database 1.2.2
Display student data 1.2.3
Update student data 1.2.4
Result of the update data 1.2.5

Figure 10. Lowest level DFD for Student input component.

3.2.2 *Component* 2 *(Edplan data input)*

The goal of this component is to provide a web interface to let students input their edplan information to the Edplan prototype database. The information students input will be used to generate an edplan report and print it. Currently students have to complete the edplan report by typing it manually and then turn in the finished edplan to their course advisor. The course advisors then have to manually enter that data to the database via STARS. Subsequently each time the students add or drop courses, they need to see their course advisor to update their edplan data. Also, when they want to view their edplan data, again they need to ask their course advisor to print the edplan data for them. This component will make the whole edplan processes easier and more efficient for both students and course advisors. The finished product should be able to help students input all the necessary information for the edplan easily and only once, and add or drop the courses when needed with minimal faculty processing. In addition to the functions described above, students will also have the abilities to view their edplan report and print it whenever they want and without having to have their course advisor do it.

3.2.2.1 Functional Decomposition of the Edplan data input component

This component also needs students to insert their last name and last four digits of their social security number as input (for authentication), if the student doesn't use the previous component's registration function. If the student does register already, the cookie will be used to identify the student. This component enables the student to input their edplan data to be forwarded to Edplan database (Eddata table). Figure 11 shows the top level (DFD) for the component.

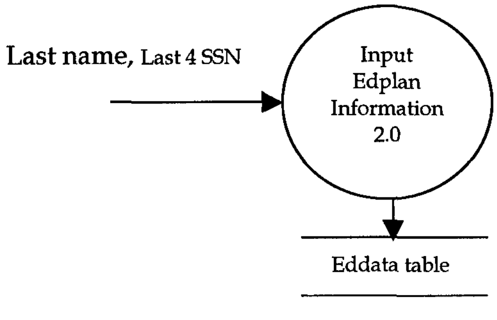


Figure 11. Level 0 DFD for Edplan input component.

The requirement for student authentication leads to the first level decomposition of the component displayed in Figure 12.

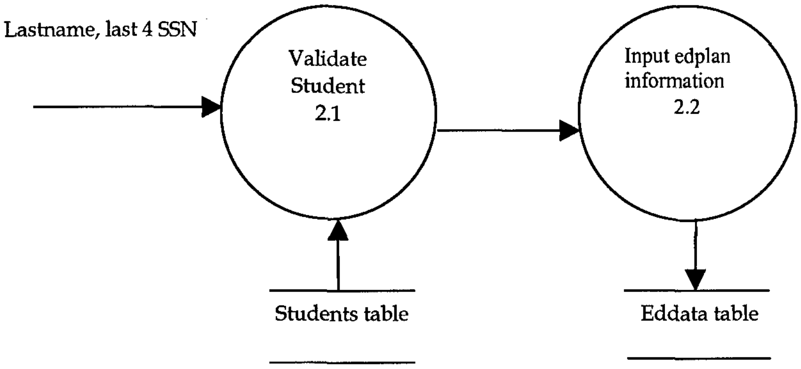


Figure 12. First level Decomposition of Edplan input component.

In order to authenticate the student to get eligible to input the edplan data, the component must query the database on the student's last name and last four of their SSN. If authenticated the component should display the edplan data input form and then insert the data into the database. Therefore, the component can be further decomposed as following (figure 13).

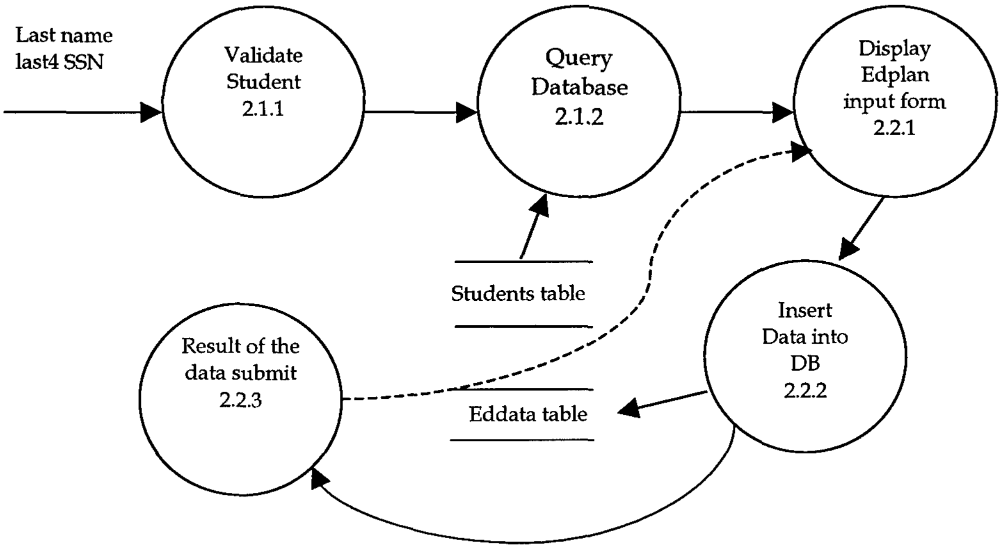


Figure 13. Lowest level Decomposition of Edplan input component.

3.2.3 *Component 3 (View and update Edplan input)*

The goal of this component is to provide ENG students with a means to view the data they input, or initiate dropping or adding courses, as needed for routine edplan maintenance. Also, a notification should be sent to their advisor (by phone or email) to official drop or add courses, from the database. There is no current means for students to directly view official edplan data by electronic means, such as over the web, and any adding or dropping courses is done exclusively by the advisor's direct action. This component can be helpful for the real system. As with the last component, students must be authenticated to ensure data security and to aid in obtaining the correct edplan data to display. The students then should have the ability to view the edplan data, and have a form to add or drop courses. Finally, the students should get a response indicating if their updates were sent successfully, and made permanent by the advisor.

3.2.3.1 Functional Decomposition of the view and update edplan component

This component uses cookies created from the student's last name and last four of their SSN when they first signed in to generate the correct data and have the access to add or drop courses. In order to view the edplan data and add or drop courses, this component must query the database and present the data to the student in an HTML form and also provide a form to add or drop courses. Figure 14 shows the DFD for the component.

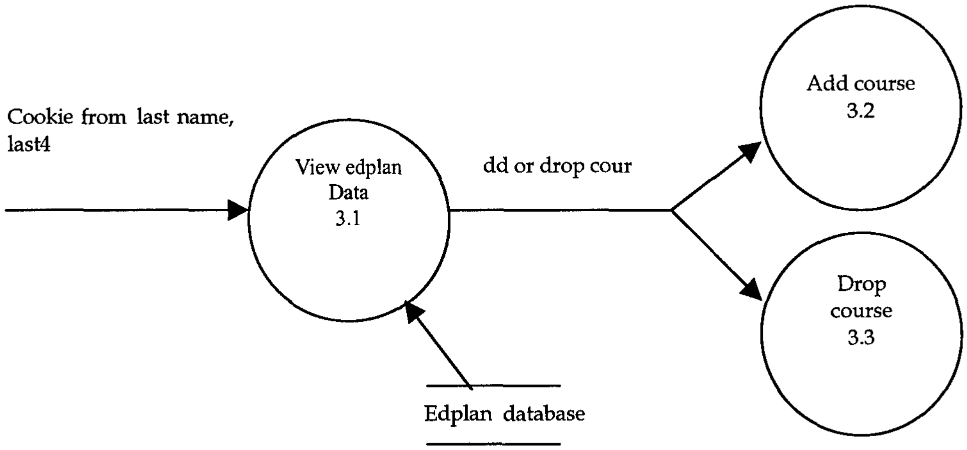


Figure 14. DFD for view and update edplan component.

To meet the requirement, the students should have a form to add or drop courses and get a response indicating if they drop or add courses successfully. The DFD can be decomposed more detailed as the figure 15.

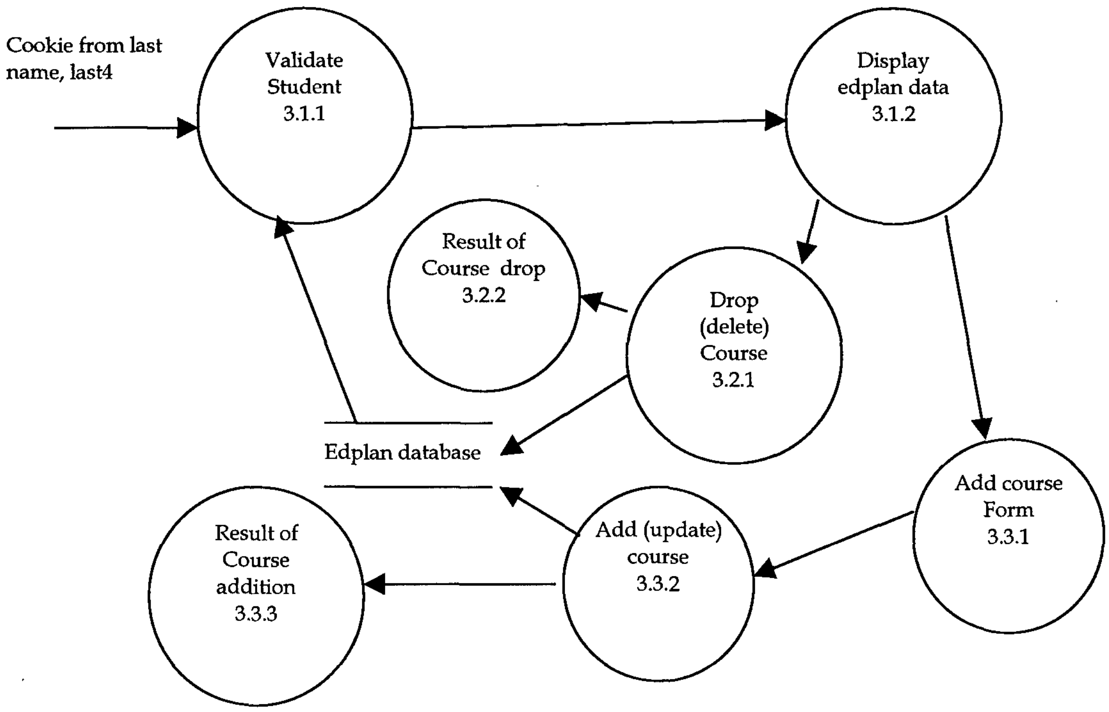

Cookie from last name, last4
Validate Student 3.1.1
Display edplan data 3.1.2
Result of Course drop 3.2.2
Drop (delete) Course 3.2.1
Edplan database
Add course Form 3.3.1
Result of Course addition 3.3.3
Add (update) course 3.3.2

Figure 15. Lowest level DFD of Edplan view and update component.

3.2.4 Component 4 (Generate Edplan checker input file)

The goal of this component is to help ENG students to generate the input file running in the edplan checker program with the data obtained from them. This component provides a means to generate the input data for students. Since the edplan checker is written in Prolog, a particular file syntax and format are expected. Thus the generated input data needs to be similar to the format of the input file as much as possible. The input data may not be generated exactly the same as the format, therefore the students might need to make some adjustment for the generated input data to be run in the edplan checker program.

Students currently need to generate the input file by hand to run against the edplan checker. This component makes it easier for them to generate the input file by using the existing data. Using the generated input file to run in the edplan checker is important to do. The edplan checker makes sure all the courses that students take satisfy the theory, system, math and sequence application requirements. Again, students must be authenticated to collect the right data. After generating the input data, students edit the data by following the instructions shown in the browser. Then students should be able to save the input data as a file and download it to their desired directory to use that file when they need to.

3.2.4.1 Function Decomposition of the Generate Edplan checker input component

This component uses the cookie to get the correct data for students and take the data source to generate the input data for edplan checker program. The generated input data then can be saved as a file and downloaded to student's machine. Figure 16 shows the top level DFD for the generate edplan checker input component.

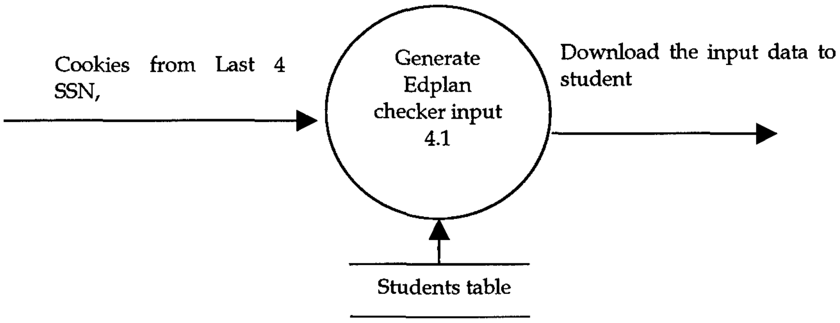

Cookies from Last 4 SSN,
Generate Edplan checker input 4.1
Download the input data to student
Students table

Figure 16. Level 0 DFD for Edplan checker input file component.

To generate the edplan checker file and download the file to the student's machine, this component can be decomposed further as the following Figure 17.

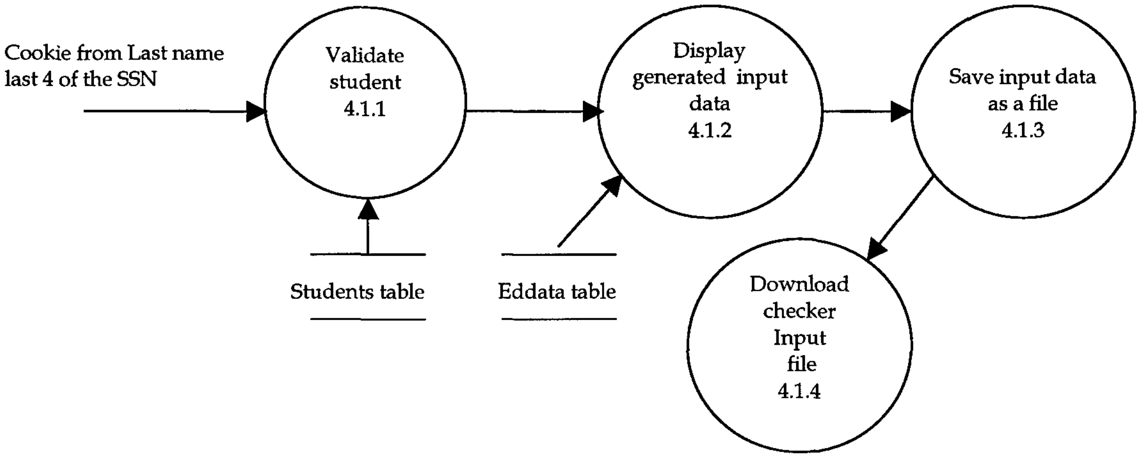

Cookie from Last name last 4 of the SSN
Validate student 4.1.1
Display generated input data 4.1.2
Save input data as a file 4.1.3
Students table
Eddata table
Download checker Input file 4.1.4

Figure 17. Lowest level DFD of the Edplan checker input component.

3.2.5 *Component 5 (Generate AFIT form 69)*

The goal of this component is to provide ENG faculty with an automated means to generate the AFIT form 69, Graduate School Of Engineering Program Summary. This component takes advantage of the data obtained from students using the data entry components to generate the AFIT form 69 for the faculty. The faculty then can use the generated form to complete the student's graduation package.

Currently, the faculty advisor has to request the student's edplan data through STARS and retype the necessary data onto the form 69 manually. Utilizing the web to generate the form is quicker and more efficient because the data was entered only once, when the student used the web interface, component 2(Edplan input component). This component helps eliminate the errors that can come from manually typing the form, and reduces the amount of time spent making corrections. This helps the faculty generate the form 69 in an easier and more efficient way.

Since this component is a function for the faculty, a login screen is required. Any faculty accessing this component must be authenticated to use this function. For this test bed, some synthetic information of the ENG faculty will be created to perform this function. The finished component should be able to retrieve student personal data, such as last name, first name, program/year and rank and associated edplan data to fill the form. Additionally, the faculty should be able to select their own thesis students from a list of students' names to generate the specific student's form 69.

3.2.5.1 Functional Decomposition of the AFIT form 69 component

This component takes the faculty's last name and last four digits of their social security number from Faculty table (also located in Edplan database) as input (authentication), then uses existing edplan data sources to generate the data for the form 69.

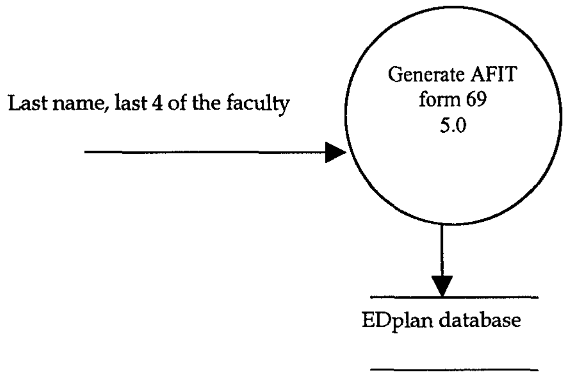


Figure 18. Level 0 DFD for AFIT form 69 component.

The requirement for faculty authentication leads to the first level decomposition of the component displayed in Figure 19.

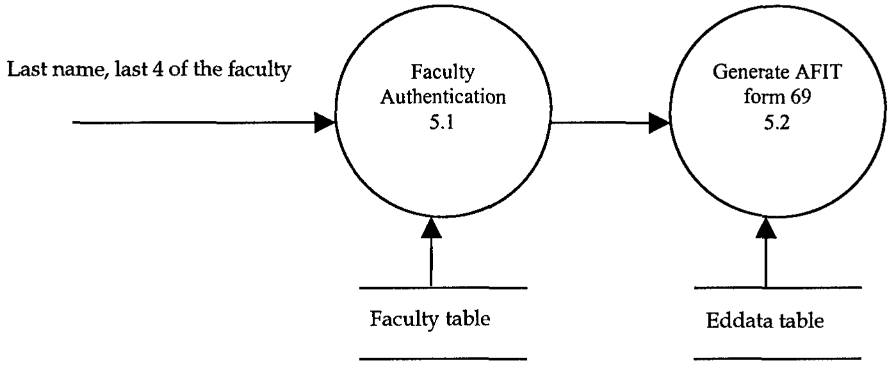

Last name, last 4 of the faculty
Faculty Authentication 5.1
Generate AFIT form 69 5.2
Faculty table
Eddata table

Figure 19. First Level DFD for AFIT form 69 component.

In order to generate the form 69 data, this component needs to query the database to obtain the necessary data from the student's database. An HTML page with a list of the faculty's students' name must be provided to be selected by the faculty for generating the specific student's form 69. Thus the component can be further decomposed to the functions shown in Figure 20.

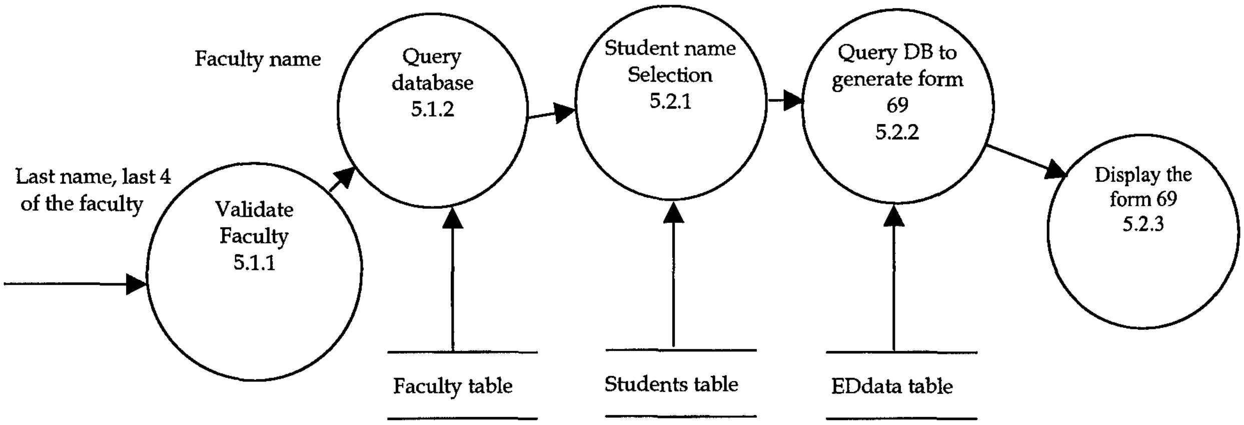

Faculty name
Query database 5.1.2
Student name Selection 5.2.1
Query DB to generate form 69 5.2.2
Last name, last 4 of the faculty
Validate Faculty 5.1.1
Display the form 69 5.2.3
Faculty table
Students table
EDdata table

Figure 20. Low level DFD of the AFIT form 69 component.

3.3 *Validation of the prototype*

The purpose of building the prototype system is to provide a service for the students and to take advantage of capability of the WWW to make edplan creation and validation easier and more efficient. The opinion and cooperation of the students is important. Some of the students will be invited to test the prototype system to see if the prototype is good or not and what needs to be improved. After the test, the students performing the testing will be asked to answer a questionnaire about the prototype system. The questions will determine what the students think of this prototype system. This student feedback will provide valuable information about the prototype and help to improve or correct the system.

This prototype system should at least meet the following requirements:

- The login function should be tested to demonstrate that it obtains the correct data.
- The system should be able to insert and update data correctly.
- The students should be able to view their edplan data, drop or add courses, and print the data report.
- The faculty should be able to generate the form 69 data for a specific student on demand.
- The prototype should be able to generate the edplan checker input data and download it to student's desired location.

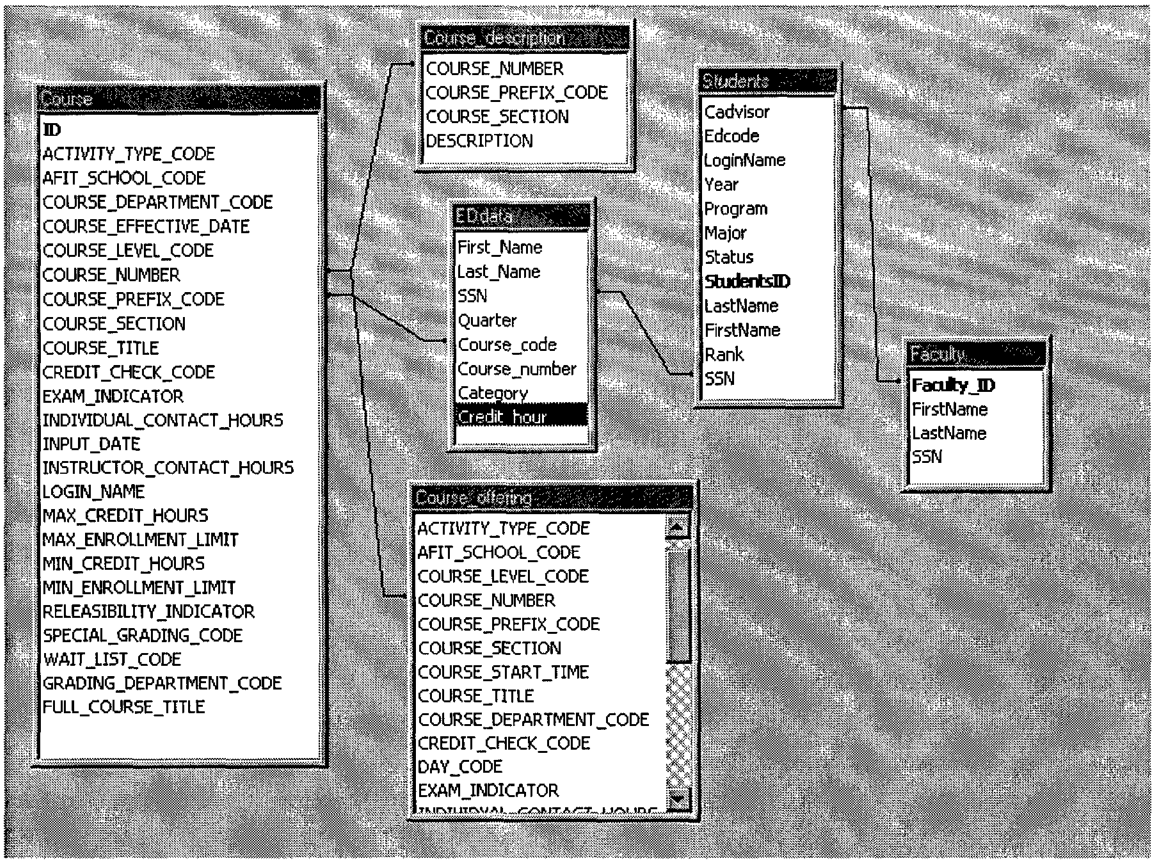
Through the students' test for the prototype system, the functions of the system will be evident. The comments from the students will help determine the success of the project.

4 Implementation and Evaluation of the prototype system

Chapter 2 provides some background information on database connectivity technologies, on the software technologies that could be used to build web database applications and on Dipiro's methodology [Dipiro98]. Chapter 3 describes using Dipiro's methodology to choose the technologies for building a prototype system and to aid in the discovery and design of the five components that comprise that prototype system. This chapter presents the implementation of those five components described in Chapter 3, and evaluates the finished product on the basis of student feedback questionnaire response.

4.1 Design Implementation of the Prototype System

This section presents the implementation for each of the five components comprising the target prototype system. The component's functional requirements are described in chapter 3. Before going through the implementation of each component, a diagram showing the tables of the Edplan database used in this prototype system is displayed in Figure 21. The Eddata, Students, and Faculty tables are populated and referenced routinely in the implementation of this prototype. The remaining tables (Course, Course_offerring and course_description) are official AFIT files and are used for reference only.



Course
ID
ACTIVITY_TYPE_CODE
AFIT_SCHOOL_CODE
COURSE_DEPARTMENT_CODE
COURSE_EFFECTIVE_DATE
COURSE_LEVEL_CODE
COURSE_NUMBER
COURSE_PREFIX_CODE
COURSE_SECTION
COURSE_TITLE
CREDIT_CHECK_CODE
EXAM_INDICATOR
INDIVIDUAL_CONTACT_HOURS
INPUT_DATE
INSTRUCTOR_CONTACT_HOURS
LOGIN_NAME
MAX_CREDIT_HOURS
MAX_ENROLLMENT_LIMIT
MIN_CREDIT_HOURS
MIN_ENROLLMENT_LIMIT
RELEASIBILITY_INDICATOR
SPECIAL_GRADING_CODE
WAIT_LIST_CODE
GRADING_DEPARTMENT_CODE
FULL_COURSE_TITLE

Course_description
COURSE_NUMBER
COURSE_PREFIX_CODE
COURSE_SECTION
DESCRIPTION

EDdata
First_Name
Last_Name
SSN
Quarter
Course_code
Course_number
Category
Credit_hour

Course_offering
ACTIVITY_TYPE_CODE
AFIT_SCHOOL_CODE
COURSE_LEVEL_CODE
COURSE_NUMBER
COURSE_PREFIX_CODE
COURSE_SECTION
COURSE_START_TIME
COURSE_TITLE
COURSE_DEPARTMENT_CODE
CREDIT_CHECK_CODE
DAY_CODE
EXAM_INDICATOR
INDIVIDUAL_CONTACT_HOURS

Students
Cadvisor
Edcode
LoginName
Year
Program
Major
Status
StudentsID
LastName
FirstName
Rank
SSN

Faculty
Faculty_ID
FirstName
LastName
SSN

Figure 21. Tables of the Edplan database used by the project.

4.1.1 Implementation of Component 1 (Student Info Input)

As described in Chapter 3, through the considerations prescribed by Dipiro's methodology, we have determined Microsoft Active Server Pages (including HTML, scripts) are the appropriate software technology to use for building the prototype system.

Based on the function designs analyzed for each component in chapter 3, each component is implemented by following the lowest level DFD provided in last chapter. The lowest level DFD of component 1 with the technologies used for each sub-component is shown in figure 22.

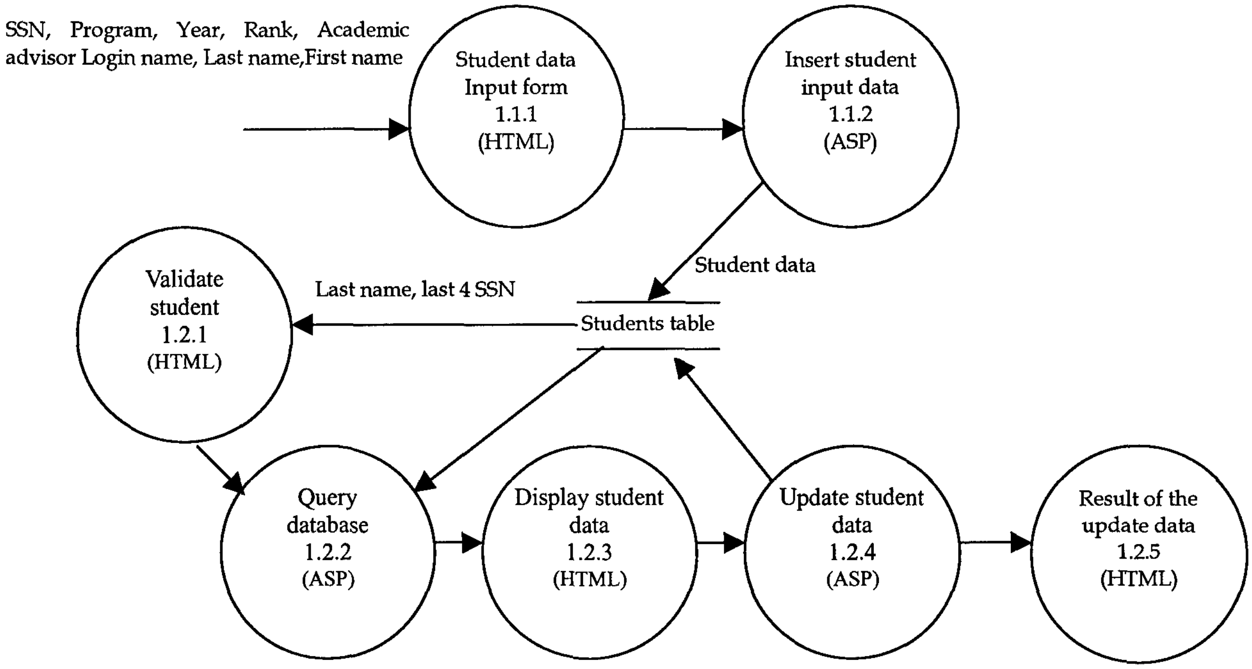
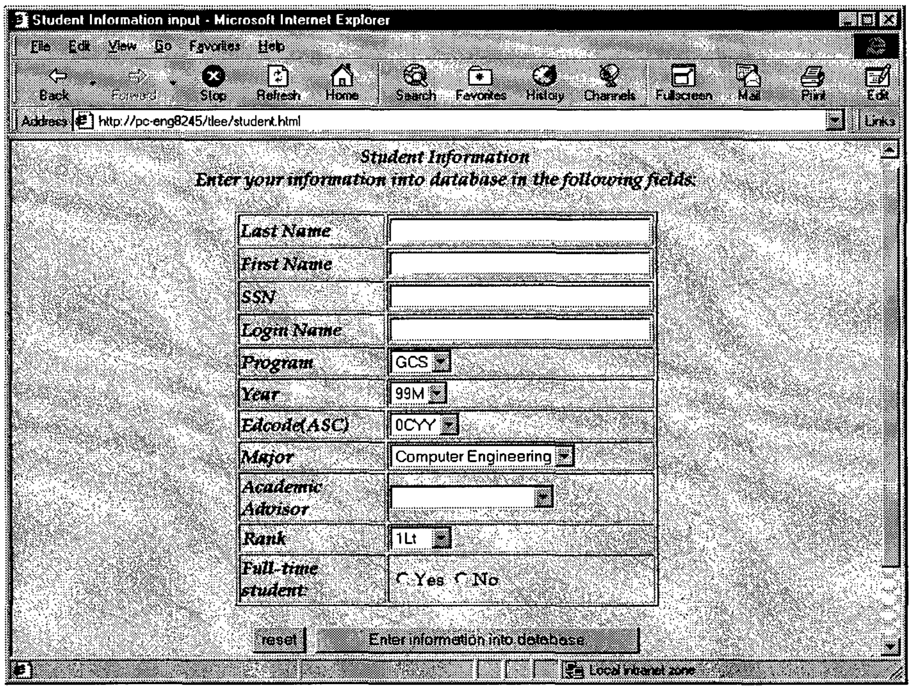

SSN, Program, Year, Rank, Academic advisor Login name, Last name,First name
Student data Input form 1.1.1 (HTML)
Insert student input data 1.1.2 (ASP)
Student data
Students table
Last name, last 4 SSN
Validate student 1.2.1 (HTML)
Query database 1.2.2 (ASP)
Display student data 1.2.3 (HTML)
Update student data 1.2.4 (ASP)
Result of the update data 1.2.5 (HTML)

Figure 22. Student input component DFD with used technologies.

This component uses HTML to build a student input form for students to input their personal data. The data they input will be important and necessary for subsequent components. To make the data input simple, several drop-down controls are used to offer selections of the correct information for students to choose from, such as Year, Program, Rank, Academic Advisor and Edcode (academic code for AFIT students). Figure 23 shows the screen shot from the implementation of sub-component 1.1.1 in Figure 22.

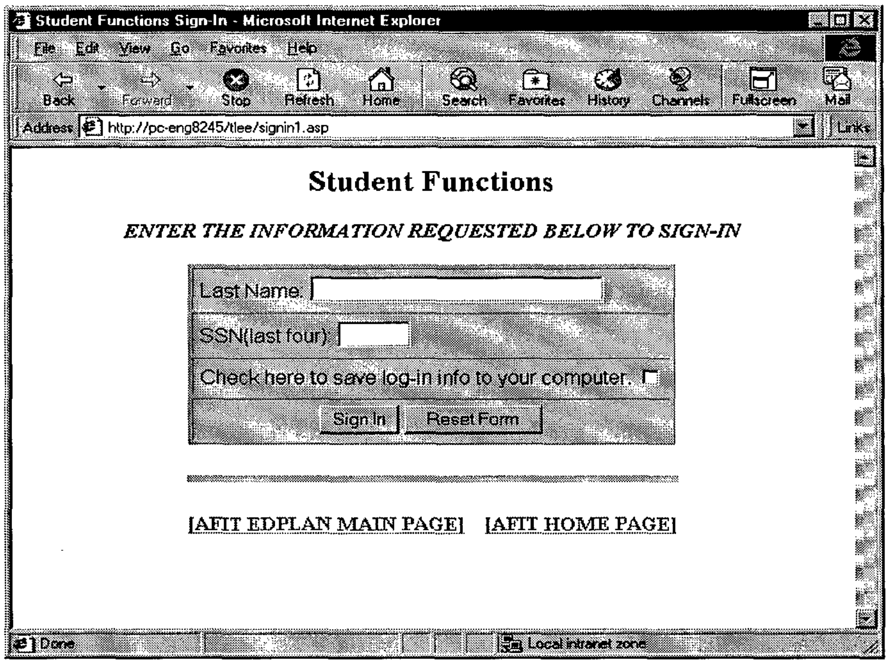

Student Information input - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit
Address http://pc-eng8245/tlee/student.html
Links
reset
Enter information into database
Local intranet zone

Student Information
Enter your information into database in the following fields:

Last Name	
First Name	
SSN	
Login Name	
Program	GCS
Year	99M
Edcode(ASC)	0CYY
Major	Computer Engineering
Academic Advisor	
Rank	1Lt
Full-time student:	○ Yes ○ No

Figure 23. Student data input component.

Sub-Component 1.1.2 is invoked by pressing the `SUBMIT` button of the Sub-Component 1.1.1 to insert the data into the student table in Edplan database. Since ASP is the chosen software technology and is very capable for accessing the database, ASP is also used to implement this sub-component. After students input their data, authentication is required before they can view or change their personal data. A static HTML page (Process 1.2.1), which allows the students to enter their last name and last four digits of their SSN, is provided for authentication. The screen shot of the authentication application is shown in Figure 22.


Student Functions Sign-In - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail
Address http://pc-eng8245/tlee/signin1.asp
Links
Done
Local intranet zone

Student Functions

ENTER THE INFORMATION REQUESTED BELOW TO SIGN-IN

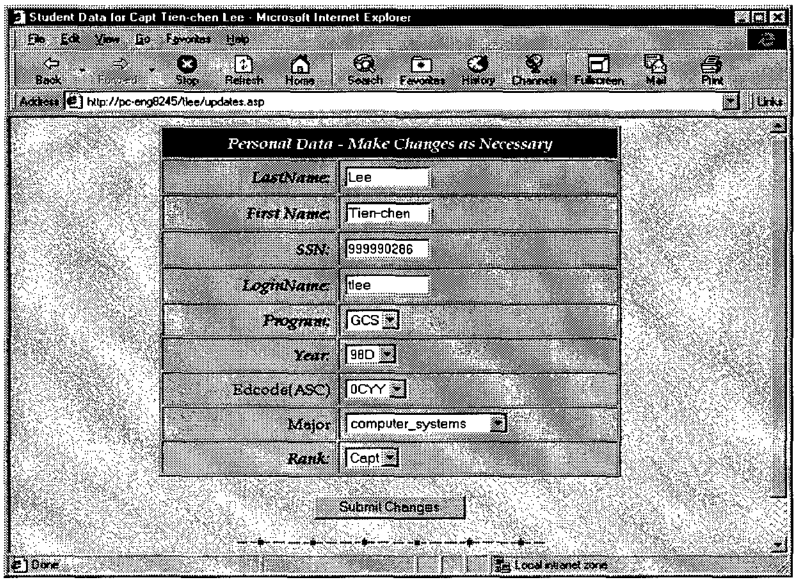
Last Name:	
SSN(last four):	
Check here to save log-in info to your computer.	□

Sign In | Reset Form

[AFIT EDPLAN MAIN PAGE] [AFIT HOME PAGE]

Figure 24. Student Identity Authentication page.

Sub-Component 1.2.2 queries the database for a record matching the last name and last four digits of SSN input by the student in sub-component 1.2.1. If a matching record is found in the database, an HTML page (process 1.2.3) with the student's personal information is then displayed (Figure 25). The student may now review and update the information as necessary.

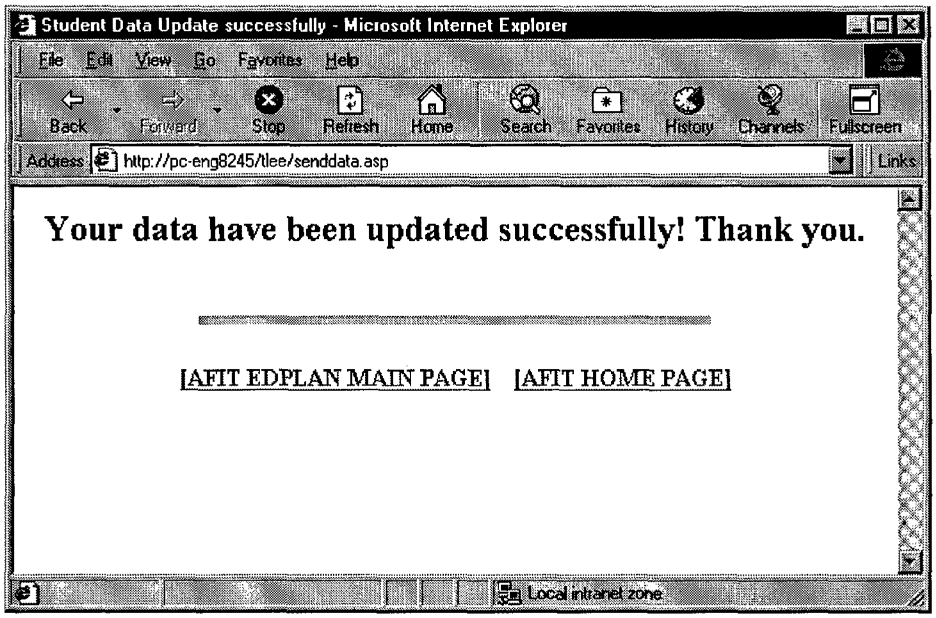

Student Data for Capt Tien-chen Lee - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Address http://pc-eng8245/tlee/updates.asp
Links
Done
Local intranet zone

Personal Data - Make Changes as Necessary	
LastName:	Lee
First Name:	Tien-chen
SSN:	999990286
LoginName:	tlee
Program:	GCS
Year:	98D
Edcode(ASC)	0CYY
Major	computer_systems
Rank:	Capt

Submit Changes

Figure 25. Student Data Display page.

If the student makes any changes and then presses the `SUBMIT CHANGES` button, sub-component 1.2.4 is invoked to update the data. Sub-Component 1.2.5 then generates the static HTML page that informs the student of the result of their submission as shown in Figure 26.


Student Data Update successfully - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back
Forward
Stop
Refresh
Home
Search
Favorites
History
Channels
Fullscreen
Links
Local intranet zone

Address http://pc-eng8245/tlee/senddata.asp

Your data have been updated successfully! Thank you.

[AFIT EDPLAN MAIN PAGE] [AFIT HOME PAGE]

Figure. 26. Result of Data Update page.

4.1.2 Implementation of the component 2 (Edplan data input)

This component is the key to the edplan administration system. It enforces security via a logon session (sub-component 2.1.1) similar to the student information function (Figure 24). The entire component is implemented through the use of two ASPs and three HTML pages. Figure 27 shows the DFD for the Edplan Input component indicating the implementation technologies.

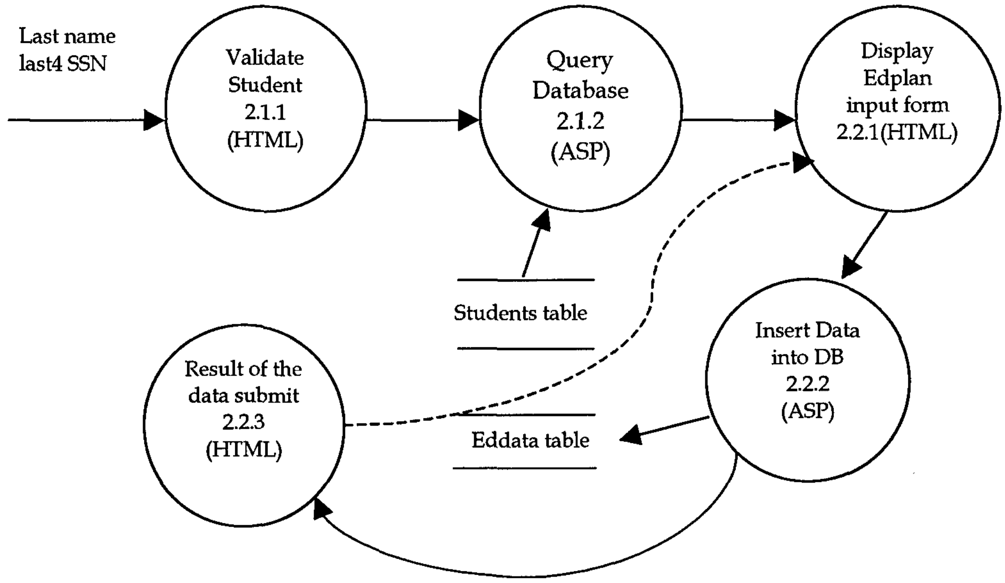
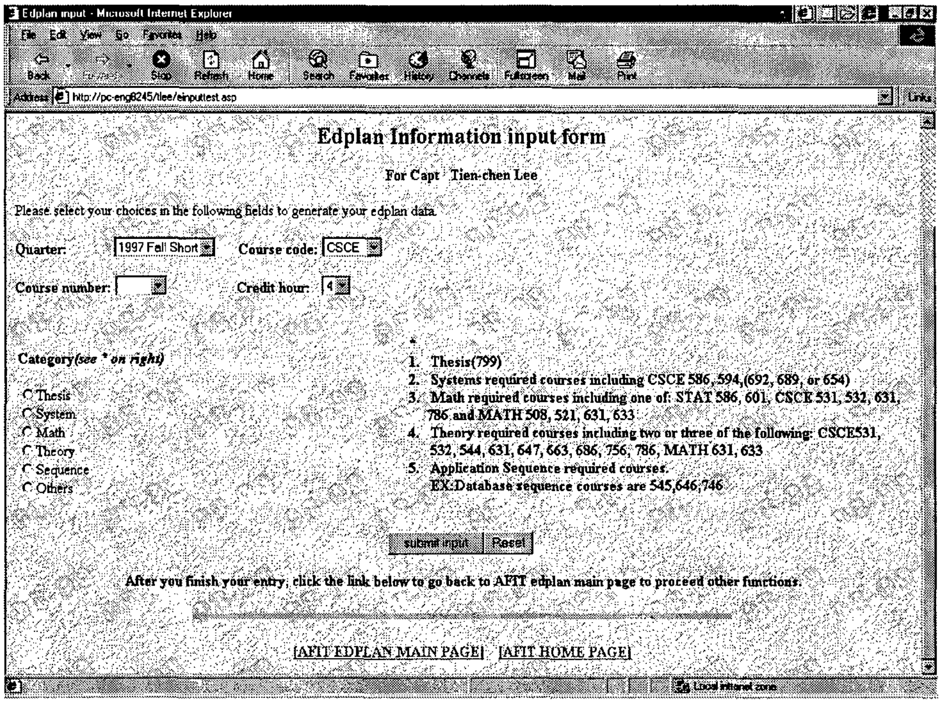

Last name
last4 SSN
Validate Student 2.1.1 (HTML)
Query Database 2.1.2 (ASP)
Display Edplan input form 2.2.1(HTML)
Students table
Insert Data into DB 2.2.2 (ASP)
Result of the data submit 2.2.3 (HTML)
Eddata table

Figure 27. Lowest level Decomposition of Edplan input.

Sub-Component 2.1.2 is implemented by ASP, and queries the database on the last name and the last four of the student to see if the student exists in the database. If the student's data is found, a static HTML page, the edplan data input form, is displayed to the student (sub-component 2.2.1). The student then enters his/her edplan data into the form. For the same reasons as for the previous component, the input form uses drop-down menus and radio boxes to make the input process easier for students. Figure 28 shows the screen shot of the sub-component 2.2.1 (Edplan Input Form)



Edplan input - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Address http://pc-eng8245/tlee/einputtest.asp
Links

Edplan Information input form

For Capt Tien-chen Lee

Please select your choices in the following fields to generate your edplan data.

Quarter: 1997 Fall Short Course code: CSCE

Course number: Credit hour: 4

Category*(see * on right)*

○ Thesis
○ System
○ Math
○ Theory
○ Sequence
○ Others

*

1. Thesis(799)
2. Systems required courses including CSCE 586, 594,(692, 689, or 654)
3. Math required courses including one of: STAT 586, 601, CSCE 531, 532, 631, 786 and MATH 508, 521, 631, 633
4. Theory required courses including two or three of the following: CSCE531, 532, 544, 631, 647, 663, 686, 756, 786, MATH 631, 633
5. Application Sequence required courses.
   EX:Database sequence courses are 545,646,746

After you finish your entry, click the link below to go back to AFIT edplan main page to proceed other functions.

[AFIT EDPLAN MAIN PAGE] [AFIT HOME PAGE]

Figure 28. Edplan Input Form page.

Once inputs/changes have been made and the SUBMIT INPUT button pressed, the input data is then sent to the second ASP (sub-component 2.2.2) which inserts the data into the Eddata table. The student will then receive a response (sub-component 2 2.3) indicating the success or failure of their submission.

4.1.3 Implementation of Component 3 (Edplan View & update)

This component allows the students to view and update the data that they previous input for their edplan. The lowest level DFD of the View and Update Edplan component indicating the technologies used is shown in the figure 29.

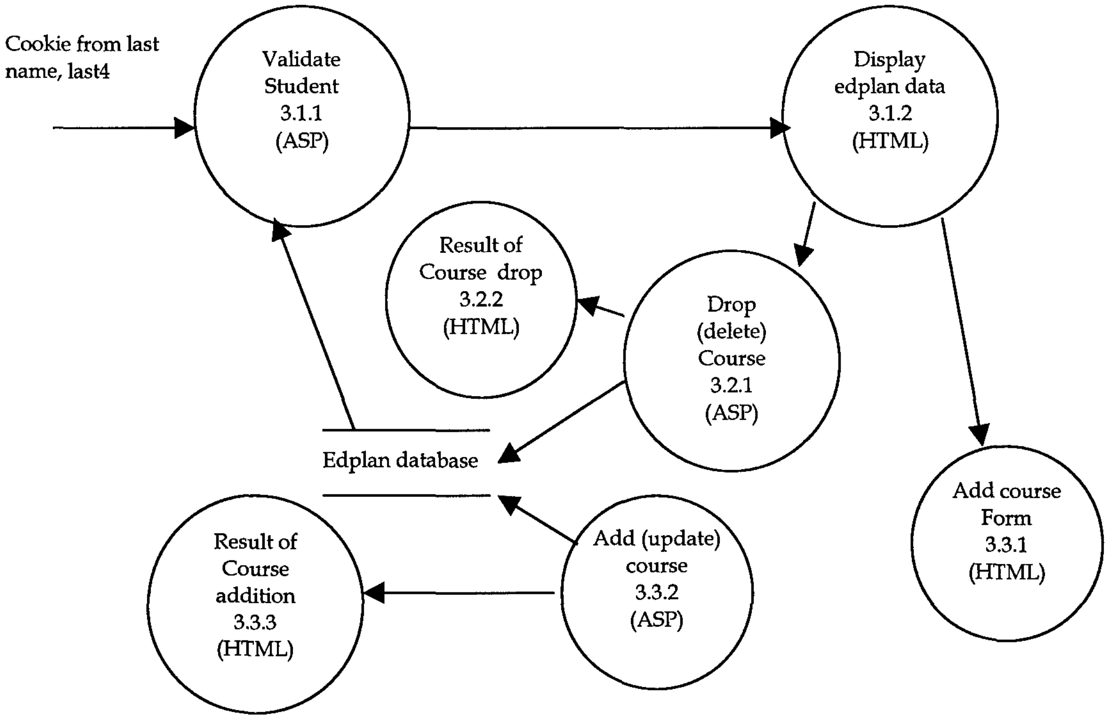
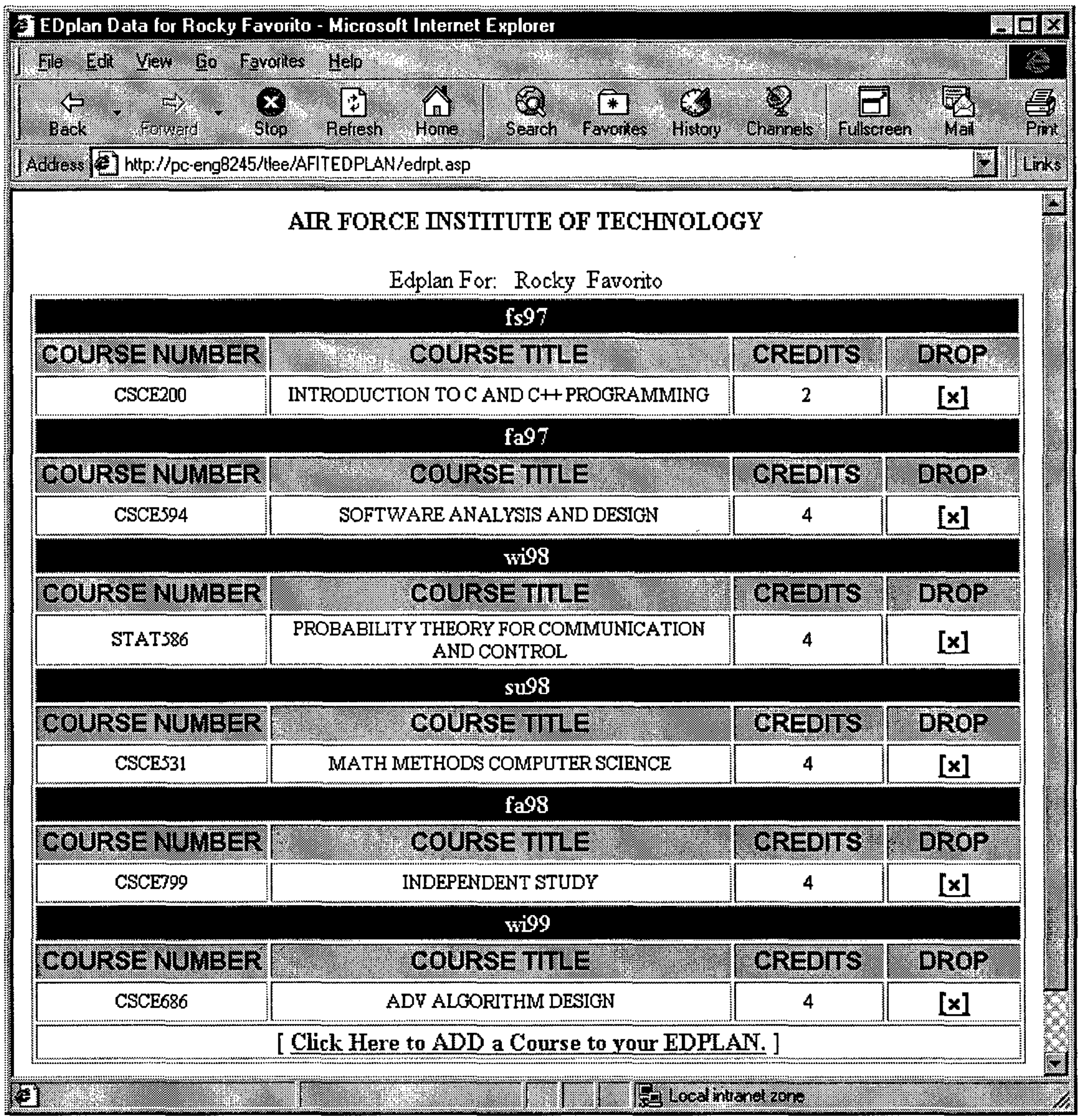

Cookie from last name, last4
Validate Student 3.1.1 (ASP)
Display edplan data 3.1.2 (HTML)
Result of Course drop 3.2.2 (HTML)
Drop (delete) Course 3.2.1 (ASP)
Edplan database
Add course Form 3.3.1 (HTML)
Result of Course addition 3.3.3 (HTML)
Add (update) course 3.3.2 (ASP)

Figure 29. Lowest level DFD of Edplan View and Update component.

To view and update the Edplan data, the students need to be authenticated as before. However, if the students want to view their edplan data, they must have already through the actions implemented as Component 2. Thus sub-component 3.1.1 uses the "cookies" generated previously to get the accurate data for the students by using ASP code. Sub-Component 3.1.2 then displays the student's edplan information with an HTML page (Figure 30).



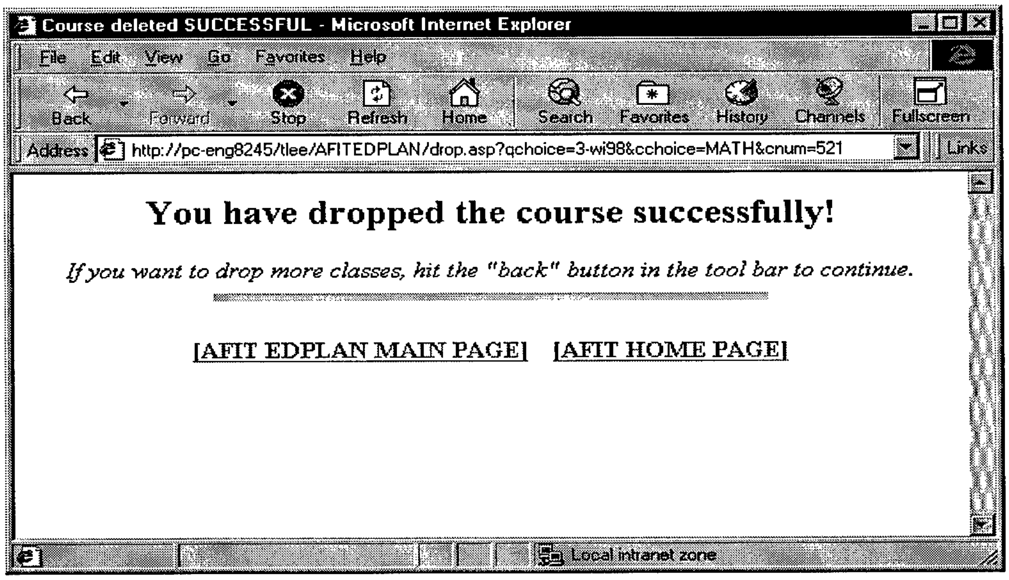
AIR FORCE INSTITUTE OF TECHNOLOGY

Edplan For: Rocky Favorito

fs97			
COURSE NUMBER	COURSE TITLE	CREDITS	DROP
CSCE200	INTRODUCTION TO C AND C++ PROGRAMMING	2	[x]
fa97			
COURSE NUMBER	COURSE TITLE	CREDITS	DROP
CSCE594	SOFTWARE ANALYSIS AND DESIGN	4	[x]
wi98			
COURSE NUMBER	COURSE TITLE	CREDITS	DROP
STAT586	PROBABILITY THEORY FOR COMMUNICATION AND CONTROL	4	[x]
su98			
COURSE NUMBER	COURSE TITLE	CREDITS	DROP
CSCE531	MATH METHODS COMPUTER SCIENCE	4	[x]
fa98			
COURSE NUMBER	COURSE TITLE	CREDITS	DROP
CSCE799	INDEPENDENT STUDY	4	[x]
wi99			
COURSE NUMBER	COURSE TITLE	CREDITS	DROP
CSCE686	ADV ALGORITHM DESIGN	4	[x]
[Click Here to ADD a Course to your EDPLAN.]			

Figure 30. The generated Edplan report page.

In this edplan data display page, there is a "cross" link (Process 3.2.1) embedded in the last column of the edplan report table after each course data and an "add course" link (3.3.1) at the end of the edplan report table. The "cross" link is used to drop the course by pressing it. This sub-component (Process 3.2.1) then uses ASP to access the database to delete the specified course. Sub-Component 3.2.2 is the HTML page showing the success of dropping the course (Figure31).


Course deleted SUCCESSFUL - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen
Address
Links
Local intranet zone

http://pc-eng8245/tlee/AFITEDPLAN/drop.asp?qchoice=3-wi98&cchoice=MATH&cnum=521

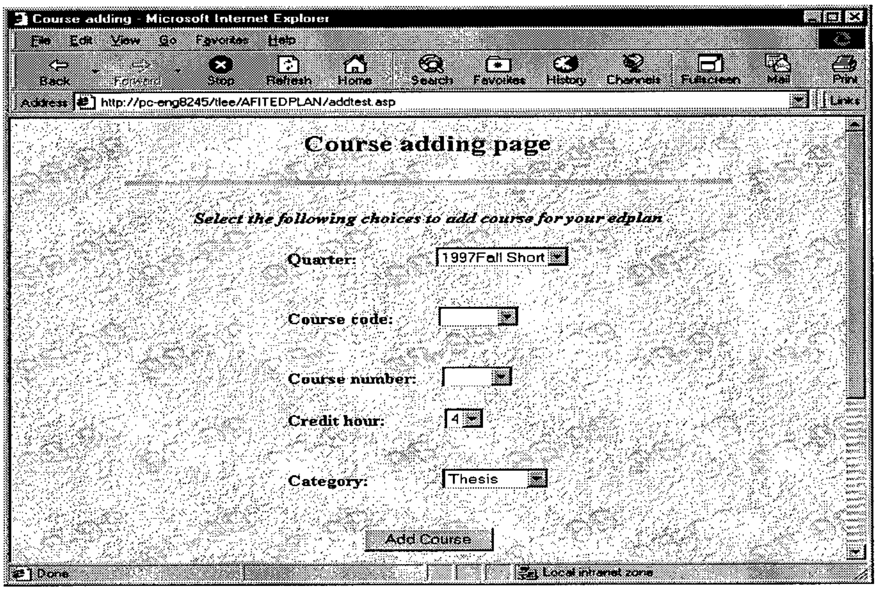
You have dropped the course successfully!

If you want to drop more classes, hit the "back" button in the tool bar to continue.

[AFIT EDPLAN MAIN PAGE] [AFIT HOME PAGE]

Figure 31. Result of dropping a course.

After pressing the ADD COURSE link, a "Course adding" HTML page (Process 3.3.1) will appear on the screen (Figure 32) to let the students add the desired course in a specific quarter. After selecting the course data and pressing the Add Course button, Process 3.3.2 executes ASP to update the data in the database. Process 3.3.3 shows the results as, depicted in Figure 32.


Course adding - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Address
Links
Done
Local intranet zone

http://pc-eng8245/tlee/AFITEDPLAN/addtest.asp

Course adding page

Select the following choices to add course for your edplan

Quarter: 1997Fall Short

Course code:

Course number:

Credit hour: 4

Category: Thesis

Add Course

Figure 32 Screen shot of the result of course dropping

4.1.4 Implementation of Component 4 (Edplan checker file)

Component 4 generates a specially encoded and formatted input file for AFIT's automated Edplan Checker program. The component also allows saving the file and downloading the file for submission to the Edplan Checker program Figure 33 shows the lowest level DFD for the Edplan Checker Input component with the technologies used for implementation.

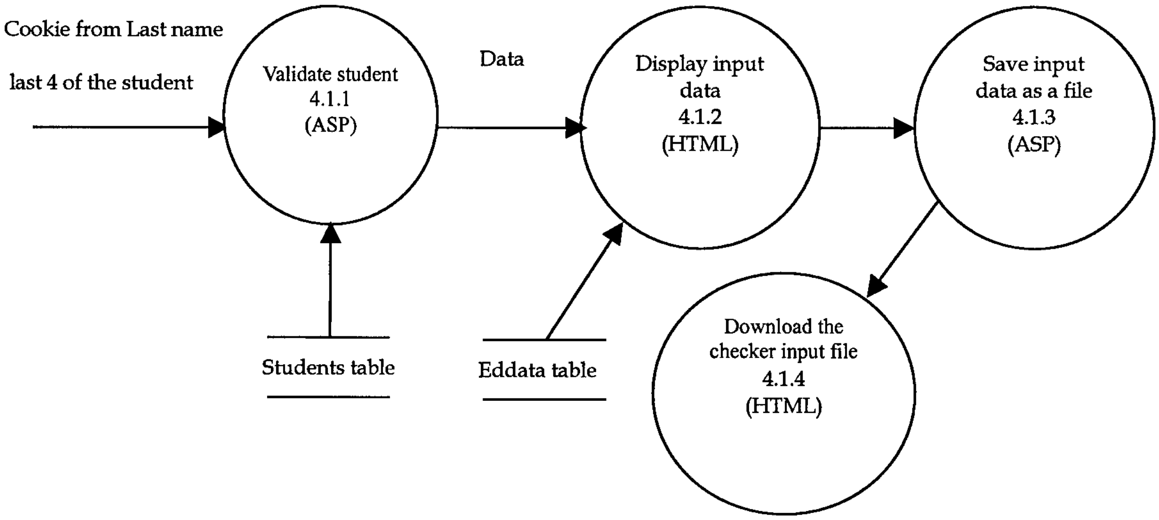
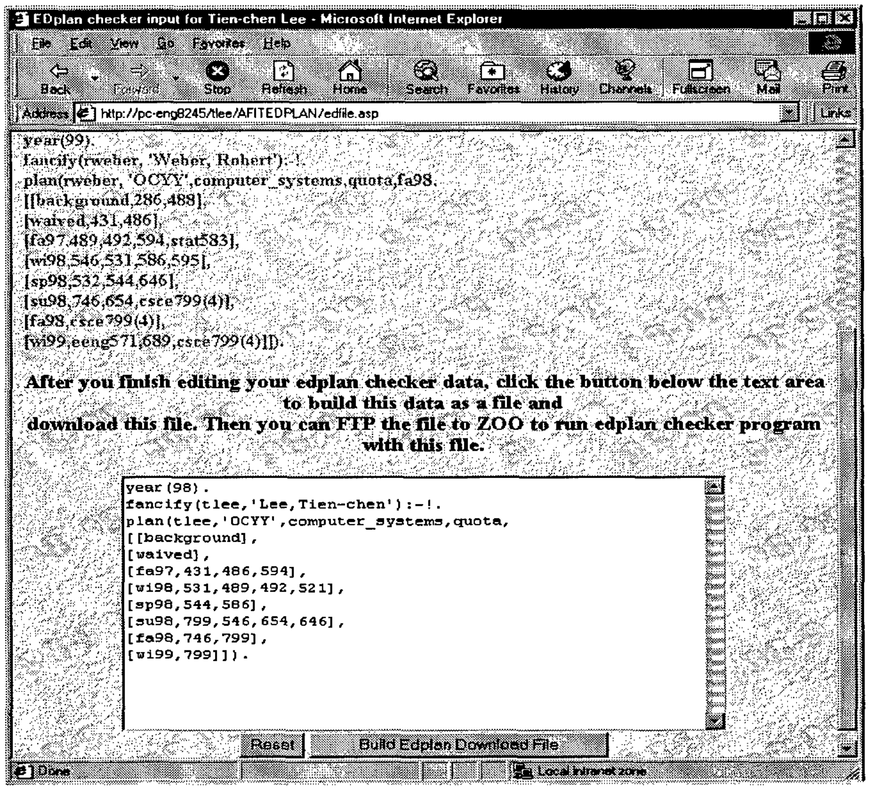


Figure 33. Lowest level DFD of the Edplan checker input component.

Sub-Component 4.1.1 used ASP code to get a student's last name and last four of their SSN stored as "cookies" generated in Component 1. This information is used for lookups into the Eddata and Students tables to retrieve the necessary course and personal data to create the edplan checker input file. Then the sub-component 4.1.2 returns an HTML page from the server with the input data in an editable text format after the ASP is executed (Figure 34).



EDplan checker input for Tien-chen Lee - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Address http://pc-eng8245/tlee/AFITEDPLAN/edfile.asp
Links

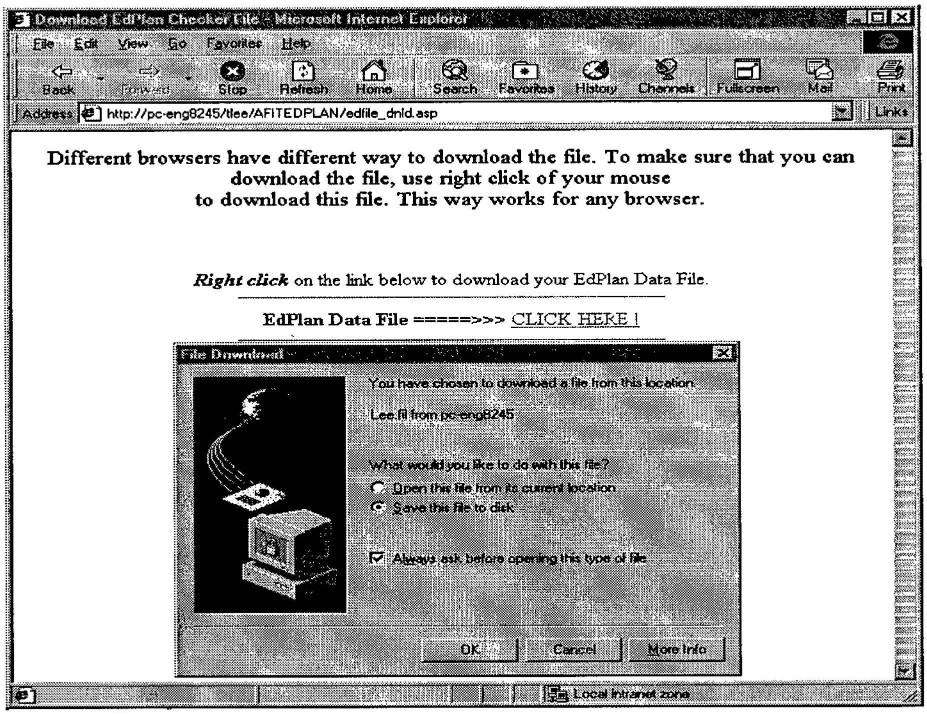
```
year(99).
fancify(rweber, 'Weber, Robert'):-!.
plan(rweber, 'OCYY',computer_systems,quota,fa98,
[[background,286,488],
[waived,431,486],
[fa97,489,492,594,stat583],
[wi98,546,531,586,595],
[sp98,532,544,646],
[su98,746,654,csce799(4)],
[fa98,csce799(4)],
[wi99,eeng571,689,csce799(4)]]).
```

After you finish editing your edplan checker data, click the button below the text area to build this data as a file and download this file. Then you can FTP the file to ZOO to run edplan checker program with this file.

```
year(98).
fancify(tlee,'Lee,Tien-chen'):-!.
plan(tlee,'OCYY',computer_systems,quota,
[[background],
[waived],
[fa97,431,486,594],
[wi98,531,489,492,521],
[sp98,544,586],
[su98,799,546,654,646],
[fa98,746,799],
[wi99,799]]).
```

Figure 34. Edplan Checker Input data generated in Component 4.

Once the input data is displayed, the students make any necessary corrections or additions. Clicking on the `Build Edplan Download File` button will invoke another ASP to save the possible edited data as a plain text file (sub-component 4.1.3). Next it displays a page (sub-component 4.1.4) form which the student may to download the input file to their local machine. The download page is shown in Figure 35. After downloading, the student may transfer the file to the server which executes the Edplan Checker application.


Download EdPlan Checker File - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Address
Links
File Download
OK
Cancel
More Info
Local intranet zone

http://pc-eng8245/tlee/AFITEDPLAN/edfile_dnld.asp

Different browsers have different way to download the file. To make sure that you can download the file, use right click of your mouse to download this file. This way works for any browser.

Right click on the link below to download your EdPlan Data File.

EdPlan Data File =====>>> CLICK HERE !

You have chosen to download a file from this location.

Lee.fil from pc-eng8245

What would you like to do with this file?

○ Open this file from its current location

● Save this file to disk

☑ Always ask before opening this type of file

Figure 35. Edplan Checker Input File download page.

4.1.5 Implementation of Component 5 (AFIT form 69 generator)

Component 5, presently the last component of this prototype system provides a faculty-specific function. This component automatically generates any specific student's completed AFIT Form 69, which has previously been a manual and error-prone task for the faculty advisors. The lowest level DFD for the AFIT Form 69 generation component is shown in Figure 36.

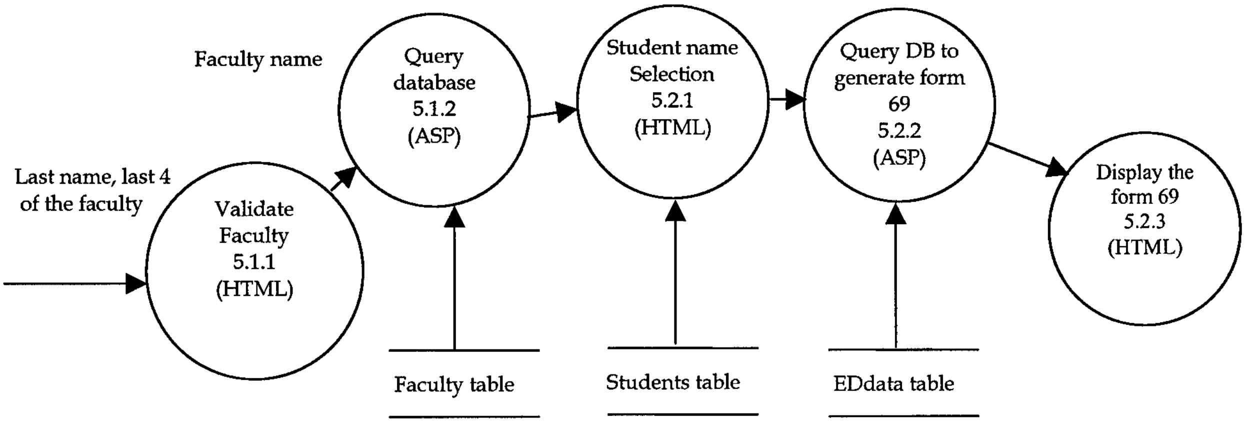
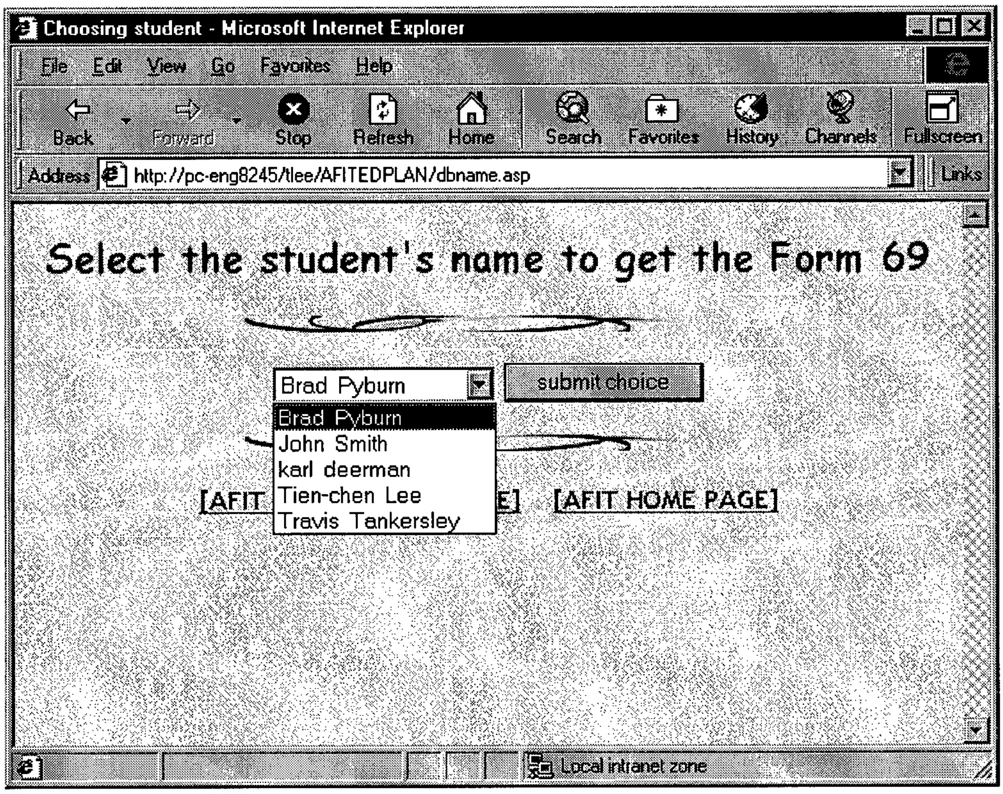

Faculty name
Last name, last 4 of the faculty
Validate Faculty 5.1.1 (HTML)
Query database 5.1.2 (ASP)
Student name Selection 5.2.1 (HTML)
Query DB to generate form 69 5.2.2 (ASP)
Display the form 69 5.2.3 (HTML)
Faculty table
Students table
EDdata table

Figure 36. Low level Decomposition of the AFIT form 69 component.

Sub-Component 5.1.1 provides a static HTML page with an authentication function (not shown). This function is similar to the students' authentication dialogue from Component 1 and 2 and requires faculty last name and last four of SSN. This component invokes sub-component 5.1.2 (ASP) to check if the faculty is an authorized user. If so, a static HTML page with a pick list of the faculty member's students' names is presented (sub-component 5.2.1). This dialogue screen lets the faculty select a student whose Form 69 is to be generated. The students' name in the pick list is based on the "Faculty Advisor" input field in sub-component 1.1.1 entered by each student. The screen shot of this page is shown in Figure 37.



http://pc-eng8245/tlee/AFITEDPLAN/dbname.asp

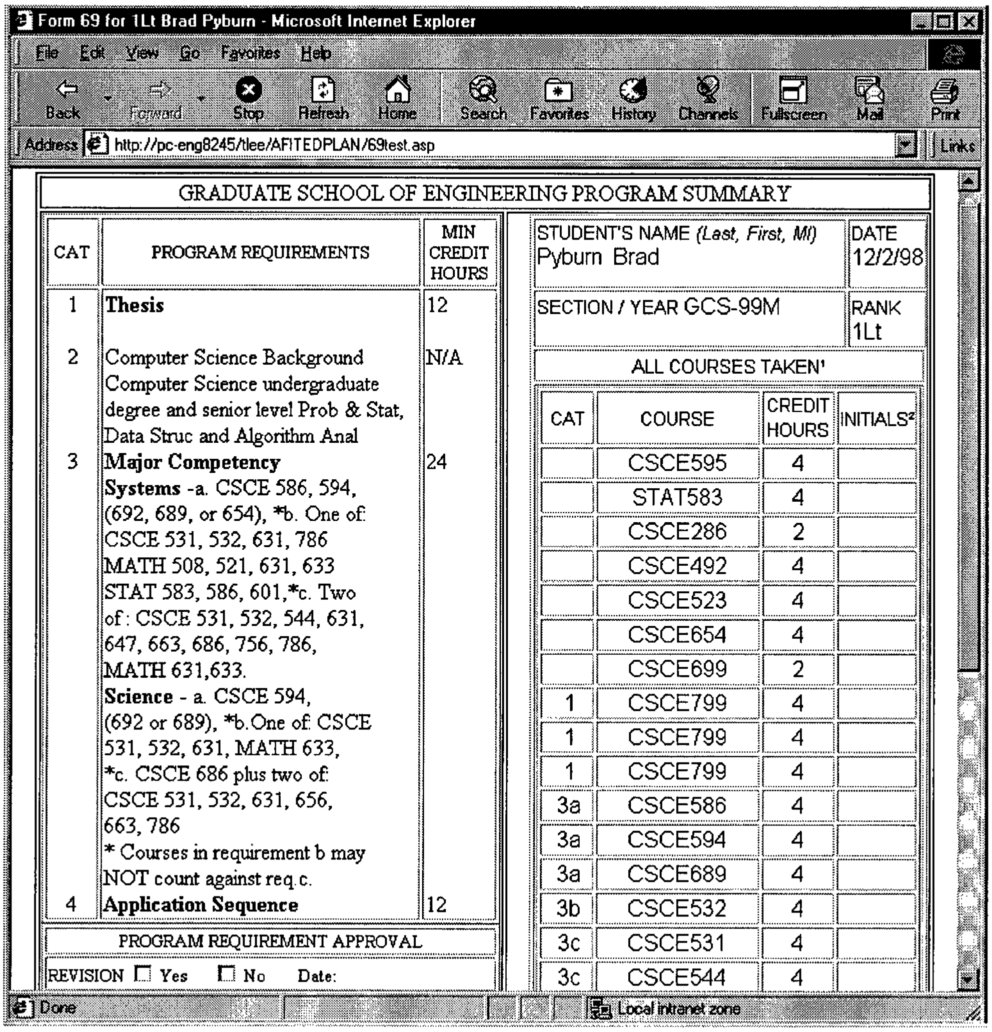
Select the student's name to get the Form 69

Brad Pyburn

- Brad Pyburn
- John Smith
- karl deerman
- Tien-chen Lee
- Travis Tankersley

Figure 37. Students' Name pick list.

Sub-Component 5.2.2 is invoked by pressing the `SUBMIT CHOICE` button after selecting the student's name. This sub-component uses the name selected to search the Eddata table to obtain the student's edplan information. A static HTML page with the data-filled Form 69 is generated and sent to the faculty in a browser format (sub-component 5.2.3). This sub-component displays the AFIT Form 69 with the information of the chosen student (Figure 38). Faculty may print the form from the browser.



Form 69 for 1Lt Brad Pyburn - Microsoft Internet Explorer

Address http://pc-eng8245/tlee/AFITEDPLAN/69test.asp

GRADUATE SCHOOL OF ENGINEERING PROGRAM SUMMARY

CAT	PROGRAM REQUIREMENTS	MIN CREDIT HOURS
1	**Thesis**	12
2	Computer Science Background Computer Science undergraduate degree and senior level Prob & Stat, Data Struc and Algorithm Anal	N/A
3	**Major Competency** **Systems** -a. CSCE 586, 594, (692, 689, or 654), *b. One of: CSCE 531, 532, 631, 786 MATH 508, 521, 631, 633 STAT 583, 586, 601,*c. Two of : CSCE 531, 532, 544, 631, 647, 663, 686, 756, 786, MATH 631,633. **Science** - a. CSCE 594, (692 or 689), *b.One of: CSCE 531, 532, 631, MATH 633, *c. CSCE 686 plus two of: CSCE 531, 532, 631, 656, 663, 786 * Courses in requirement b may NOT count against req.c.	24
4	**Application Sequence**	12

PROGRAM REQUIREMENT APPROVAL

REVISION □ Yes □ No Date:

STUDENT'S NAME *(Last, First, MI)* Pyburn Brad	DATE 12/2/98
SECTION / YEAR GCS-99M	RANK 1Lt

ALL COURSES TAKEN[1]

CAT	COURSE	CREDIT HOURS	INITIALS[2]
	CSCE595	4	
	STAT583	4	
	CSCE286	2	
	CSCE492	4	
	CSCE523	4	
	CSCE654	4	
	CSCE699	2	
1	CSCE799	4	
1	CSCE799	4	
1	CSCE799	4	
3a	CSCE586	4	
3a	CSCE594	4	
3a	CSCE689	4	
3b	CSCE532	4	
3c	CSCE531	4	
3c	CSCE544	4	

Figure 38. Screen Shot of the Generated AFIT Form 69.

4.2 Evaluation of the prototype system

The complete prototype system is composed of both a student academic function (Edplan and Edplan checker input file generator) and a faculty function (Form 69 generator). This section evaluates the finished prototype system by examining the comments and feedback from students and faculty against the minimum target requirements set in Chapter 3. To get the students' and faculty's opinions and feedback, on-line questionnaires are provided. The on-line questionnaires are designed to make it easy for students and faculty to input their opinions and impressions of the functions offered by this prototype system, and of course to make gathering meaningful statistics both easy and accurate.

4.2.1 Evaluation of student functions

The questionnaire for students includes ten questions. Those questions focus on the comparison of this prototype system with the previous manual "system". As described in Chapter 3, some students, specifically the AFIT/ENG GCS 99M class, were invited to test this prototype web-based edplan administration system and provide an evaluation

Class GCS 99M consists of 28 students. There were actually 14 people from the class who participated in the testing of this prototype system. Although not all of the 99M students took part in the experiment, the responses of those students who did will be considered the majority opinion regarding this prototype system. The results indicate that most of the students who have tried this web-based edplan system like this process for handling their edplan administration. According to student comments, most of them think this web-based system should be the way for future AFIT edplan administration.

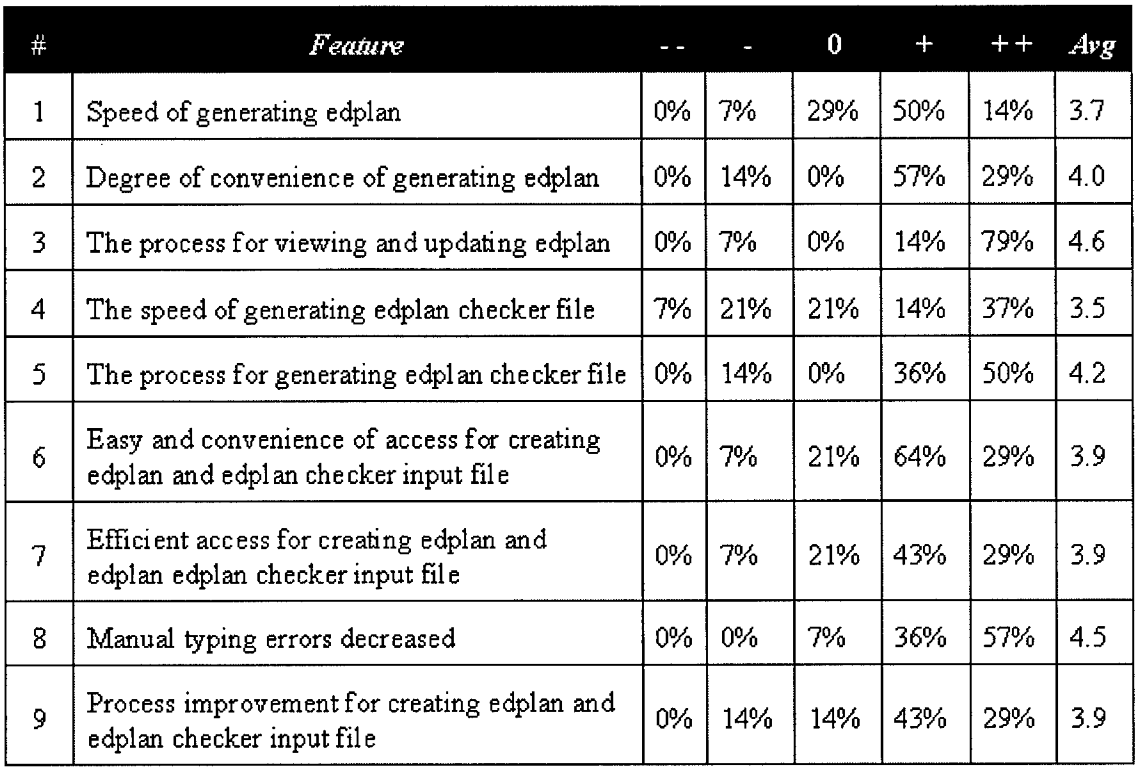
The results of the evaluation of the student's function, as derived from the questionnaires, are provided in Table 1. In this table, there are eight columns. The first column "#" represents the question number and the second column, "Features", shows the features compared for improvement over the previous version for each question in this questionnaire. The columns from the third to the seventh column, have column titles as "- -", "-", "0", "+", "+ +". Those columns show the statistics information based on the responses from students for each question. The following expresses what those columns stand for respectively:

- The third column "- -" represents answers to a question that stress the prototype system is "far worse" than the previous version or expectation.
- The fourth column "-" represents answers to a question that stress the prototype system is "worse" than the previous or expectation.
- The fifth column "0" represents answers to a question that stress the prototype system is "the same" compared with the previous version.
- The sixth column "+" represents answers to a question that stress the prototype system is "better" than the previous version.
- The seven column "+ +" represents answers to a question that stress the prototype system is "much better" than the previous version.

- The eighth column " Ave " shows the average score for each question (highest score is "5").

The following is Table 1, which shows the statistics for each question in the questionnaire.

Table 1. The Statistics from the on-line Questionnaire



#	*Feature*	--	-	0	+	++	*Avg*
1	Speed of generating edplan	0%	7%	29%	50%	14%	3.7
2	Degree of convenience of generating edplan	0%	14%	0%	57%	29%	4.0
3	The process for viewing and updating edplan	0%	7%	0%	14%	79%	4.6
4	The speed of generating edplan checker file	7%	21%	21%	14%	37%	3.5
5	The process for generating edplan checker file	0%	14%	0%	36%	50%	4.2
6	Easy and convenience of access for creating edplan and edplan checker input file	0%	7%	21%	64%	29%	3.9
7	Efficient access for creating edplan and edplan edplan checker input file	0%	7%	21%	43%	29%	3.9
8	Manual typing errors decreased	0%	0%	7%	36%	57%	4.5
9	Process improvement for creating edplan and edplan checker input file	0%	14%	14%	43%	29%	3.9

The information generated in Table 1 clearly expresses what students think about this system. The average scores indicate that all the features that this prototype system focuses on, are acceptable and appreciated by most of the students. According to the data in Table 1, almost all the students like the web-based edplan viewing and updating function and the web access to the edplan administration. This table provides a valuable baseline evaluation for gauging the value added by future improvement.

4.2.2 Evaluation of the Faculty Function

For the faculty's function evaluation, three faculty members who were current or previous academic advisors were invited to test the web-based AFIT Form 69 generation. All these faculty members think this function largely improves the current method. They really think this application provides an efficient and convenient way of generating Form 69, decreases the chance of manually typing errors and saves time for processing the Form 69 in the current method.

Overall, using students' input data to generate the Form 69 makes the process automatically and efficient. The existence of this function really boosts the value of the whole prototype system. It shows that the data students input are reused frequently, not only to generate the edplan report, and edplan checker input file, but also to generate the Form 69 for the faculty. In short, this function validates the importance of this prototype system.

4.2.3 Overall Evaluation of the Prototype System

According to student and faculty feedback and comments, this prototype system offers many good functions, and is really an improvement over the current manual process. Except for some valid concerns about the web-interface design itself, most of the functions offered by this prototype system work as they are designed to. This prototype system does satisfy the requirements set in Chapter 3. The data that students and faculty generate are all correct, and students can update and input information without errors. The students can view and print their edplan report with no problems (most of the students like the display of the edplan report) and they can also generate the edplan checker data easily with this prototype system. Finally, this system allows the faculty to choose a specific student and generate and print the PC-version AFIT Form 69.

This prototype system achieves every main target function. The functions within the target range of this project all work correctly. Some aspects of the design of the interface of the system need to be improved, but in terms of functionality, this project can be considered complete and successful.

4.3 Summary

In this chapter, the five components of this prototype system are implemented with the chosen software technologies ASP and HTML. Screen shots of this prototype system were displayed to reveal the current state of the prototype and the DFD with the implemented technologies were provided to give the reader a complete picture and context. This chapter evaluated the finished project by examining student and faculty feedback. The results of this evaluation show this prototype system can be considered a success.

Chapter five presents the findings and conclusions based on the evaluations of the prototype system, and provides a brief evaluation of Dipiro's methodology. Finally the next chapter addresses some areas for future work.

5 *Findings and Conclusions*

The main goal of this thesis is to design and build a future AFIT web-based Education plan administration prototype system by implementing the methodology proposed by Dipiro [Dipiro98]. A secondary goal of this thesis is also to validate Dipiro's methodology by using it for this real-world target application.

Chapter 2 provides background information in the areas covered by this thesis, with focus on the data connectivity technologies, such as CGI, ODBC, JDBC, and ADO; web software technologies, such as Applet and ASP; and also the details of Dipiro's methodology. Chapter 3 follows the methodology to choose the most appropriate software technologies and design the components comprising the goal prototype system. Chapter 4 provides the implementation of the goal project and also evaluates the finished product based on student and faculty feedback after their test use of this prototype system.

5.1 Findings

This section begins with an overview of the findings made during this research and follows with some specific issues surrounding the application development and the technologies that were applied.

5.1.1 Findings from Using Dipiro's methodology

In using Dipiro's methodology for building the target project, several observations are made. The first step of Dipiro's methodology is very good in helping the developers make decisions for choosing the right technologies to use. The second helps the designers analyze and design the functions for their project. The third assists in implementing the design of the project. By using Dipiro's methodology, the range of choices among the many current technologies was narrowed down smartly. This in turn helped reduce this designer's time required for choosing the appropriate technologies, and left more time to focus on the design of this project. Also, this methodology laid a structured logic path for analyzing and designing the goal application, which gives this designer the best and most correct direction to follow.

By following the first step of the methodology, Microsoft Active Server Pages is selected as the most appropriate software technology for implementing the target project. Using ASP is also good for the size of the client pool in this goal project, since it didn't overload the server or processor. However, while this application, due to its homogeneity, runs well and efficiently in PCs with Internet Explorer 4.0 as a browser, it does not run well in the Netscape 4.0 browser (a more detailed explanation is described in section 5.1.2.2). Additionally, ASP is easy to learn which reduces the time for the designer to become familiar with this tool and facile with its constructs. Overall, ASP is considered the best choice for this case project based on the analysis of the development environment. Dipiro's methodology has been found to be complete and thorough and genuinely helped arrive at the appropriate design for this application. As long as PCs are still common platforms, the methodology is good to use for web-based data access applications for the near future.

5.1.2 *Issues with the finished prototype system*

This section describes some problems with the finished product, after the students had tested the prototype system. Each subsection covers a specific problem. Several problems were found during the testing of this system.

5.1.2.1 Data retrieval problem

When the students tested the prototype system, the problem that most students encountered was with the "special study" courses. After the students had input their chosen courses, they were able to view their edplan input on demand. If a student had chosen one or more special study courses, CSCE 699, when inputting their edplan, the edplan view page in their edplan report listed every possible CSCE 699 course in the database. This was because in the Course table, the course code for all special study courses is CSCE 699, for all instances of the course, for every student. A recommended solution is described in section 5.2.1.

5.1.2.2 Variations in web browsers

The incompatibilities between different browsers and even client versions can affect the way a component functions or is displayed on the client's machine. For example, Microsoft Internet Explorer 3.0 doesn't support a "checker" function, which is part of the authentication function code. If a student used a PC with IE 3.0 and tried to use the prototype system, the browser would raise an error message that the authentication function was invalid and would not allow the student to proceed with the other functions. Fortunately, most computers in AFIT use IE 4.0 and will do so more in the near future, so this didn't affect this prototype system substantially. Using the Netscape browser caused another problem which was a malfunction in reading the "cookies" generated by the prototype system. Netscape couldn't read the "cookies" created from the authentication function correctly. It could still read the "cookies", but just couldn't use them in the way that it is supposed to. Additionally, the way that Netscape displays HTML-encoded information, such as list boxes and check boxes, is very different from how an Internet Explorer. browser does this.

5.2 Recommendations

Based on the knowledge gained from the background research and from designing and developing several web software components for the target project, the following recommendations are offered with regard to the "Data

Retrieval" problems described in section 5.1.2.1. Additionally recommendations for the future database model for web-based application are provided.

5.2.1 Recommended Solution for Course "CSCE 699" Problem

CSCE 699 is the course code for all "special study" courses in GCE/GCS program. All the "special study" courses in the Course table have the same course code and course number, but different course titles. Thus, when students include CSCE 699 on the edplan input page, all "special study" courses are selected and displayed on the edplan report page. To solve this problem, a specific method is needed to deal with the "special study" courses. The recommended solution is to create a separate CSCE 699 table for tracking every.699 course taken in any quarter by any student. Any edplan report would require the inclusion of appropriate course recorded in this table. The Edplan Input Form page (sub-component 2.2.1) shown in Figure 28 should have a input area for students to type in the title of the special study course taken. This method should solve this "special study" course problem.

5.2.2 Database Model for Web-based Application

According to Dipiro's conclusion chapter and after experiencing developing and designing the components for the prototype system, relational databases are found to be efficient and appropriate for a web-based data access project.

The way that the relational database is designed works out well with this type of application. A relational database is highly recommended for use for the future web-based database application. Additionally, it would be optimal if the Oracle relational database or Microsoft SQL server can be used. These powerful relational databases would make the application robust, efficient and ultimately more robust in a multiple thread operating environment, where multiple users access the system concurrently. The Microsoft Access database used for this research is robust only for a single user.

5.3 Conclusions

By using Dipiro's methodology, it helps to choose the appropriate technology based on the analysis of the development environment, and to develop the components for the target project. For the PC-centric, Microsoft-dominant near future, it is still a good methodology to implement for web-based data access applications.

As revealed in the evaluation, the finished prototype edplan administration system is not yet "shrink wrap" quality, however it clearly presents an improvement over the current manual method. Additionally, it provides a very good foundation for future AFIT edplan administration. According to the students' comments, and current trends, a web-based system such as this will be the desired platform for the future.

5.4 *Future work*

Many areas exist for future work based on the results of this prototype system. Generally speaking, the interface design is not felt to be as precise or intuitive as it could have been due to the time constraints for its development, and some functions still have room for more improvement. Some of those functions include the edplan input page, which only allows the student to input one course at a time (the greatest time consuming part in this system); and the edplan checker data generation function, which still needs to add some information, (such as thesis quarter, background, and waived courses) and also needs to FTP the file to run on ZOO (a unix-based network of workstations in AFIT/ENG) to get the output. Therefore, more effort can be directed at improving these functions so that they operate more efficiently, and are transparent and completely web-based.

Since the prototype system developed in this thesis was not robust across all browsers of different versions and software platforms, it would be advisable to build a more portable system. Since portability is the very issue that most of the applications are pursuing, this should be considered a key next improvement.

Finally, due to the popularity and convenience of web access, similar academic data administration applications should be developed on this platform to make the current processes easier and more efficient.

Bibliography

[Dipiro98] Dipiro, Daniel L. " Methodology for the Analysis and Design of Internet Software Components Providing Relational Database Access Through the World Wide Web" Thesis, Dayton: Air Force Institute of Technology, 1998.

[FastCGI] "FastCGI." *WWWeb,* http://www.fastcgi.com/

[Hettih97] Hettihewa, Sanjaya and Kelly Held. " Teach yourself Active Server Pages in 14 Days" 1997.

[IIS 4.0] "Internet Information Server 4.0", WWWeb, http://www.microsoft.com/iis/guide/

[JAVA96] Hamann Jerrid. "*Java, an Object-Oriented Programming*" 8 Sep 96. WWWeb, http://www.cs.tamu.edu/people/jhamann/jdbc/report/node5.html

[Jen96] Jen, Howard. "The OpenPath RDA/ODBC Driver" A White paper WWWeb http://www.openpath.com/products/odbcpaper.htm 1996.

[Litwin98] Litwin, Paul. "ADO programming" Microsoft Office Conference, http://www.microsoft.com/accessdev/articles/moves202.htm 25 Sep 98.

[Muell97] Mueller, John Paul, ActiveX from the Ground Up, Berkeley: Osborne, 1997.

[North96] North, Ken. "Building Web Databases" *7 Aug 96 WWWeb,* http://www.webtechniques.com/features/jan97/weekly/012497/north.shtml

[North97] North, Ken "How do data access APIs work." *16 May 97, WWWeb, http://ourworld.compuserve.com/homepages/Ken_North/dataacce.htm*

[Rumba91] Rumbaugh, James; Blaha, Michael; Premerlani, William; Eddy, Frederick; Lorensen, William. "Object Oriented Modeling and Design." Englewood Cliffs: Prentice Hall, 1991.

[SUN96] North, Ken "Sun's JDBC spec turns up database heat under Java" PC Week Labs May 8, 1996.

Vita

Captain Tien-Chen Lee was born on 27 October 1970 in Kaohisung, Taiwan. He graduated from the Chung Chang armed forces preparatory school (high school) in 1989 and graduated from the Chinese Air Force Academy (CAFA) in 1993. He was selected to remain at the CAFA as a teacher assistant in the Aeronautics Department immediately upon graduation. After being a teacher assistant for two years, he was selected to attend the Air Force Institute of Technology. Upon graduation from AFIT, Capt. Lee will rejoin the faculty of the CAFA.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 10 December 1998	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE

A Web-based Prototype for AFIT Edplan Administration

5. FUNDING NUMBERS

6. AUTHOR(S)

Tien-Chen Lee, Captain, Taiwan AF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
2950 P Street
WPAFB, OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCS/ENG/98D-02

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFIT/RRD
Mr. Randall Baker
2950 P Street
WPAFB, OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This document details the design, development, and evaluation of a prototype course registration and reporting system for the students and faculty of the United States Air Force Institute of Technology. The web-based system provides HTML-based client interfaces and Active Server Page server processes for interaction with the relational databases used to manage course and personnel data. The system prototype was developed following the "Engineering Software Components for Web-Database Access" methodology of Dipiro. A survey of modern web-based database access techniques is first provided and Dipiro's methodology is reviewed as background. The remainder of the document details the application of the methodology as a decision aid for decomposing system requirements into a series of user interaction and data access functions. Then, again following the methodology, an analysis of extant web-database access techniques is performed in the search for the most appropriate one. Next, the developed prototype system's functions are described and depicted via screen capture images. Finally the results of prototype evaluation via user feedback surveys are provided along with recommendations for future system improvement. Ultimately, this work stands as a validating test case for the Dipiro's methodology.

14. SUBJECT TERMS

Databases, Internet, Object Oriented Programming, Software Engineering

15. NUMBER OF PAGES

100

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL